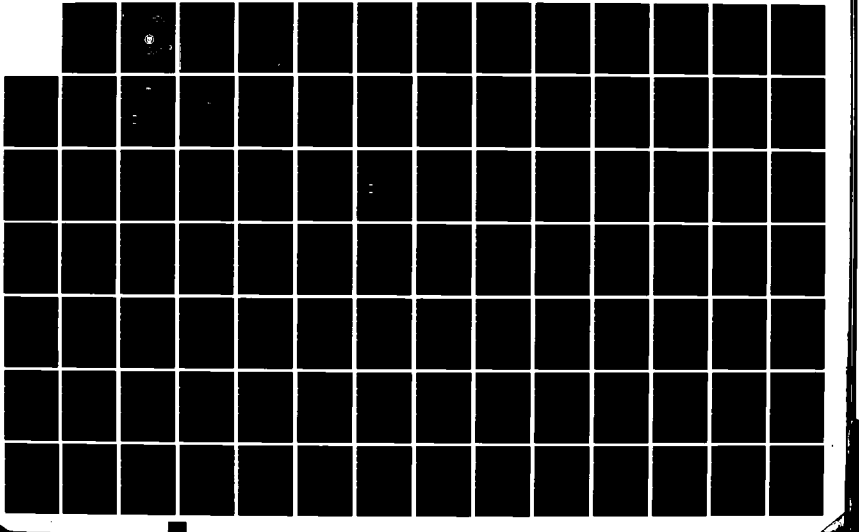
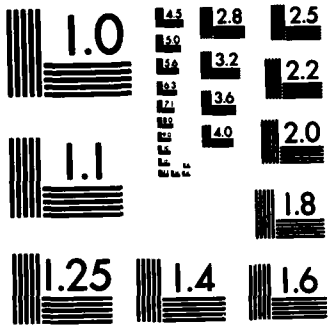
AD-A123 225 ECONOMETRIC MODEL FOR OPTIMIZING TROOP DINING FACILITY OPERATIONS USER'S A..(U) ARMY CONCEPTS ANALYSIS AGENCY BETHESDA MD A C MANGUSO DEC 82 CAA-D-82-4 1/2

UNCLASSIFIED F/G 8/6 NL





DOCUMENTATION
CAA-D-82-4

ECONOMETRIC MODEL FOR OPTIMIZING TROOP DINING FACILITY OPERATIONS

USER'S AND PROGRAMER'S REFERENCE MANUAL

DECEMBER 1982

PREPARED BY
ANALYSIS SUPPORT DIRECTORATE
US ARMY CONCEPTS ANALYSIS AGENCY
8120 WOODMONT AVENUE
BETHESDA, MARYLAND 20814

DISCLAIMER

The findings of this report are not to be construed as an official Department of the Army position, policy, or decision unless so designated by other official documentation. Comments or suggestions should be addressed to:

Director
US Army Concepts Analysis Agency
ATTN: CSCA-AS
8120 Woodmont Avenue
Bethesda, MD 20814

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: CAA-D-82-4 — 2. GOVT ACCESSION NO.: AD-A123225 — 3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): Econometric Model for Optimizing Troop Dining Facility Operations (User's and Programer's Reference Manual) (U)	5. TYPE OF REPORT & PERIOD COVERED: Documentation, 1 Nov 81 to 31 Oct 82
	6. PERFORMING ORG. REPORT NUMBER: CAA-D-82-4
7. AUTHOR(s): CPT August C. Manguso	8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: US Army Concepts Analysis Agency, 8120 Woodmont Avenue (ATTN: CSCA-ASA), Bethesda, MD 20814	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of the Deputy Chief of Staff for Logistics, Department of the Army (ATTN: DALO-TST), Washington, DC 20310	12. REPORT DATE: December 1982
	13. NUMBER OF PAGES: 168
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office):	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

(U) Active Army; (U) Logistics; (U) Planning; (U) Model Development; (U) Menu Planning; (U) Goal Programing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This manual provides the user and the programer with reference material concerning the operations and maintenance of the Econometric Model for Optimizing Troop Dining Facility Operations. This model, also referred to as the menu planning model, was presented in the US Army Concepts Analysis Agency (CAA) study report CAA-SR-82-10, dated October 1982. The model is centered around a goal programing model, capable of selectively achieving goals for food cost, labor cost, acceptability and nutrition in the design of the Army Master Menu. This manual is divided into two main parts: a user's guide, and a programer's

OVER

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

reference manual. The user's guide is oriented to the non-technical user, while the programer's reference is directed to the knowledgeable FORTRAN programer. Material presented in this manual applies to that revision of the model that was placed into operations on the Burroughs BG800 computer used by the US Army Troop Support Agency (TSA) at Ft. Lee, VA. Runstreams pertaining to the revision of the model that was developed on the UNIVAC 1100/82 at CAA are included in an appendix of this manual.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

DOCUMENTATION
CAA-D-82-4

ECONOMETRIC MODEL FOR OPTIMIZING TROOP DINING FACILITY OPERATIONS

USER'S AND PROGRAMER'S REFERENCE MANUAL

December 1982

Prepared by

Analysis Support Directorate

US Army Concepts Analysis Agency
8120 Woodmont Avenue
Bethesda, Maryland 20814

CONTENTS

FIGURES

TABLES

USER'S AND PROGRAMER'S REFERENCE MANUAL

CHAPTER 1

INTRODUCTION

1-1. PURPOSE. This manual is intended to provide the user and the programer with information concerning the application and maintenance of the Econometric Model for Optimizing Troop Dining Facility Operations. The model is intended to provide a design tool which can be used to guide the analytical preparation of the Army Master Menu, and therefore the model will be referred to as the Menu Planning Model. This manual is divided into two main parts: the first, covered in Chapter 2, is a guide to the user, and the second, covered in Chapter 3, is a reference manual for the programer. The user should not have to refer to the second part in order to successfully use the model.

1-2. BACKGROUND. A detailed discussion of all the factors that went into the design of the Menu Planning Model is contained in CAA Study Report 82-10. Background information is provided here for the sole purpose of orienting the user and programer to the model design and the factors that influenced that design. Effective application of the model is largely dependent on the understanding that the user has of the model and its capabilities; therefore, the user is encouraged to read the study report before attempting to apply the model.

a. The Army Master Menu. The Army Master Menu is an integral part of the Army food program and essentially a list of "what is to be made." The Master Menu is currently published on a monthly basis and is used in the planning of meal selections. The food service sergeant may follow the Master Menu or make whatever deviations are necessary to satisfy the eating habits and desires of the soldiers eating in his dining facility. The Master Menu is important as a guide because although deviations may be allowed, the Master Menu provides for nutritional adequacy, relative cost efficiency, and general acceptability. Currently, the Master Menu is based on a 42-day menu cycle and reflects an effort to balance the factors of cost, nutrition, and acceptability.

b. Problem. The current method of developing the Master Menu is based on manual and partially automated procedures. There is no consistent analytical approach to the consideration of food cost, acceptability, nor nutritional adequacy in the design of the Master Menu. In addition, the relative labor costs of alternative menu plans are not considered.

c. Objectives. The specific objectives of the study under which the Menu Planning Model was developed are discussed in the study report. In general, those objectives provide for the development of both a methodology which provides for the consistent analytical consideration of food cost, labor cost, nutrition, and acceptability in the design of the Army Master Menu, and a model that is capable of applying that methodology. Inherent in the development of the methodology is the collection and analysis of data for recipes and selected menus, along with the identification of appropriate goals for food cost, labor cost, acceptability, and nutritional adequacy.

1-3. METHODOLOGY

a. Goal Programing (GP). GP is a mathematical programing technique that allows for the consideration of several goals in the optimization process. In this model, the goals shown in Figure 1-1 are prioritized at the time the menu is being planned. The GP algorithm then selects the combination of menus that results in the least deviation from those prioritized goals. The GP methodology employed in the Menu Planning Model is based on an attempt to achieve each goal in a preemptive fashion. Prioritizing the four goals implies that one is preferred to another, which is preferred to another, etc. Preemptive prioritization implies that one is preemptively, or infinitely, preferred to another. This means that the second priority goal may be achieved only so long as its achievement does not reduce the achievement of the first priority goal. The relative achievement of a goal is measured by the positive or negative deviation from that goal. From this it can be seen that the aim in GP is to minimize the deviation from the various goals, and with conflicting goals, the reordering of priorities can lead to entirely different solutions.

b. Menu Attributes. The Menu Planning Model incorporates a process for assessing the relative worth of menus in terms of attributes for food cost, labor cost, acceptability, and nutritional content. This process allows for the consistent analytical determination of menu attributes which are subsequently used as input to the GP algorithm.

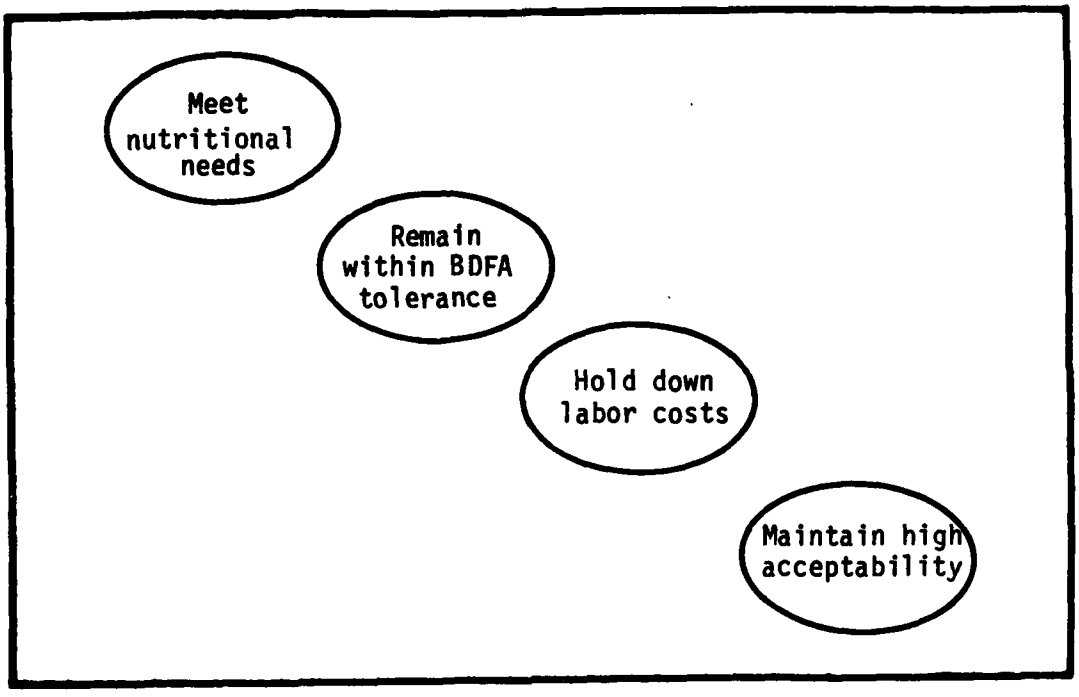


Figure 1-1. Menu Planning Goals

1-4. SYSTEM DESIGN AND MODEL STRUCTURE

a. Integration With the Existing System. The system design was intended to correspond with the logical sequence of operations in menu planning. The user is able to interface with recipe and menu data files in order to maintain and update that data. The user may set upper limits on the number of times that any menu may be repeated during the menu cycle and may subsequently alter those limits for specific menus. The user also has the capability of establishing goals and reordering priorities in order to assess the effect of those factors on the overall menu plan.

b. Modularity. The Menu Planning Model is divided into three distinct modules that correspond to the functional processes of menu planning: data maintenance, establishment of planning parameters, and generation of complete plans. They are, respectively, the data handling module, the parameterization module, and the solution module.

c. Portability. The Menu Planning Model was developed on a UNIVAC 1100/82 at the Concepts Analysis Agency, however a version has been placed on the Burroughs B6800 at Ft Lee, VA. Although there is little difference between the two versions, the instructions contained in this manual are intended to refer to the Burroughs version. Pertinent UNIVAC runstreams and file names are listed in Appendix C.

d. Data Sources. Data for the model may come from a number of sources, including a Management Information System that is to be developed at TSA. The data necessary for the successful use of the model consists of two fairly simple data files: one containing a list of recipes and the worth of those recipes in terms of food cost, labor cost, acceptability, and nutritional content for 10 nutrients; the other being a list of candidate menus and the recipes that comprise those menus. A detailed description of the data files is included in this manual.

1-5. SYSTEM APPLICATION

a. Users. It is intended that the users of the Menu Planning Model will ultimately be the TSA menu planners. The model may also have application in the assessment of concepts in menu design, but the user interfaces are intended to allow the model to be run by persons who may not be computer oriented.

b. Uses. As mentioned above, the model may have applications in assessment of different concepts, but the model is intended to be a to for use in guiding the analytical design of the Army Master Menu.

CHAPTER 2

USER'S GUIDE

2-1. MODEL STRUCTURE. The structure of the Menu Planning Model is represented schematically in Figure 2-1. This figure is not intended to be a logical flow chart since much of the detail is intentionally eliminated. The user may find it handy to refer to Figure 2-1 while using this guide in order to keep track of the interactions of the various model components. The model is actually divided into three distinct modules: the data handling module, the parameterization module, and the solution module. The remainder of this chapter discusses each module in turn. The discussion is intended to explain processes and alternative courses of action available to the user. Where necessary, data sources and input requirements will be discussed. For details on the programs and subroutines themselves, the user may wish to refer to Chapter 3, or the documented source code listing.

2-2. USER INTERACTIONS. In operating this model from a remote terminal, the user will be prompted for a variety of input. In many cases, the reply will be a simple "yes", "no", or the selection of a single digit number. In some cases, a special response must be made such as the entry of names or numbers. In those cases, the user will usually be told how to respond. If told to respond with an "F" format, that simply means to enter a real number (a number with a decimal point, i.e., 45.0). An "I" format means an integer number (no decimal, such as 45), and an "A" format means any series of numbers or characters. If a specific length, or placement of input is expected, the user will be given a display such as:

```
ENTER RECIPE NUMBER:
AAAAAAAAAA
```

This means that the user should enter up to 10 characters directly below the "As." As an example:

```
ENTER RECIPE NUMBER:
AAAAAAAAAA
L-150-1
```

Alternatively, the user may be expected to respond with mixed data, such as in this case:

```
ENTER NEW BOUND IN THE FOLLOWING FORMAT:
AA FFFFFFFFFFF
```

The response might be:

```
ENTER NEW BOUND IN THE FOLLOWING FORMAT:
```

In those cases when a specific format is not indicated, the format of the expected response is either obvious or immaterial.

2-3. DATA HANDLING MODULE

a. Purpose. The data handling module, also referred to as the data module, allows the user to interface with the basic data files. The module is represented in Figure 2-2. The user would run this module in order to perform those actions necessary to establish a valid data set. The user must ensure that the data is valid before going on to planning menus.

b. Data Sources. Data concerning recipes and menus may come from a variety of sources, including a management information system. The module maintains two data files: the recipe attribute file, and the menu component file. A third file, the menu attribute file, is generated from the data contained in the first two files. Data from other data sources may be loaded into the recipe attribute file and the menu component file.

c. Input Requirements

(1) Recipe Attribute File. This file is maintained by the data module and should not be altered by using a text editor. The data in this file and the menu component file may be modified by exercising the options provided in the data module. The format of the recipe attribute file is shown in Chapter 3. Basically, the file consists of data for each recipe, including recipe number, name, kind, food cost, labor cost, acceptability, and nutritional content for the following 10 nutrients:

Calories
Protein
Fat
Calcium
Iron
Vitamin A
Thiamin
Riboflavin
Niacin
Vitamin C

The file name is CAA/RECIPEDATA.

(2) Menu Component File. This file is also maintained by the data module and should not be altered by using a text editor. The format of the menu component file is shown in Chapter 3. This file consists of a list of selected menu numbers. These are candidate menus which may or may not be selected for inclusion in the solution. Each is defined in terms of the recipes that comprise the menu. The file name is CAA/MENU-DATA.

(3) Menu Attribute File. This file is created by the data module and should not be altered by using a text editor. The format of the menu attribute file is shown in Chapter 3. This file consists of data for each menu listed in the menu component file. This data includes the menu number, food cost, labor cost, acceptability, and nutritional content for each of the 10 nutrients mentioned earlier. The file name is CAA/MENATTDAT.

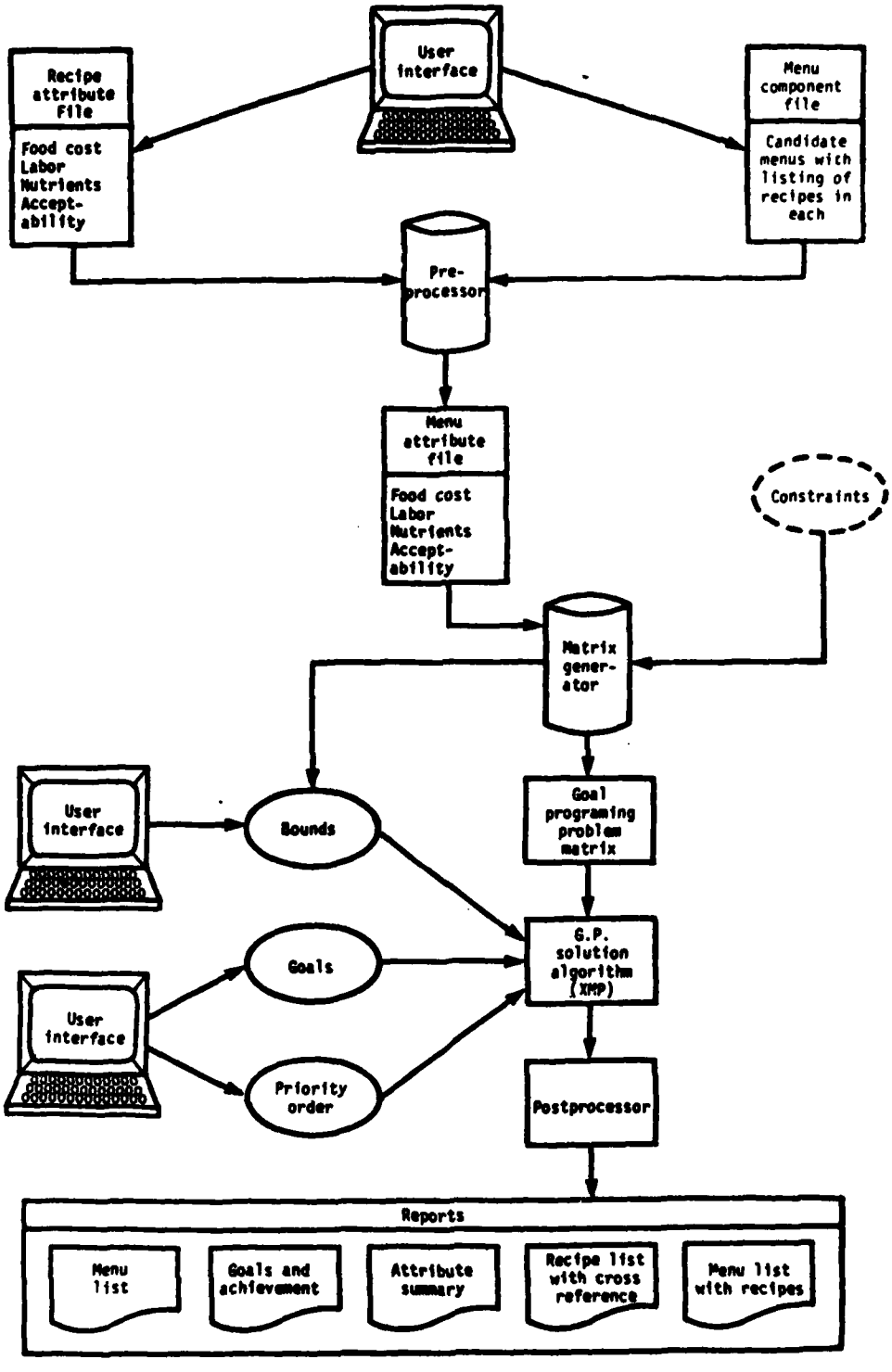


Figure 2-1. Menu Planning Model Structure

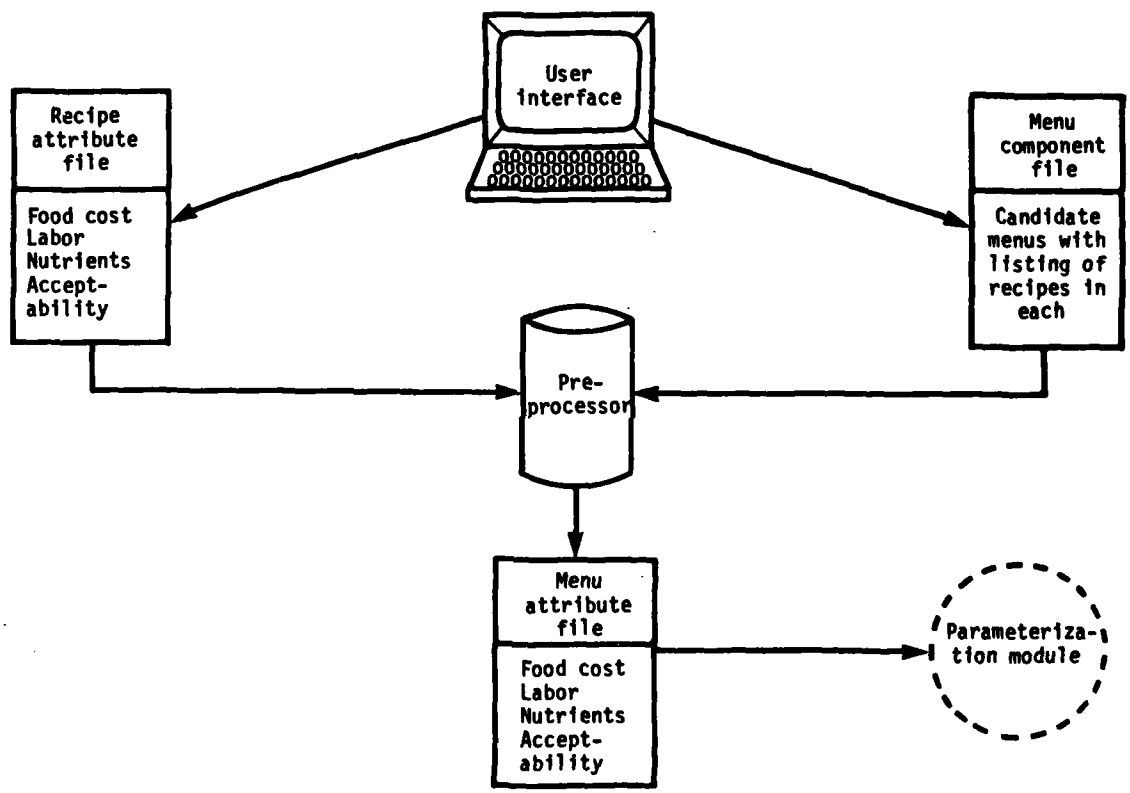


Figure 2-2. Data Handling Module

(4) External Data Files. Two classes of external data files are intended as input to the data module. The first class may be referred to as working files. They have the same format as the files maintained by the data module. In fact, once the recipe attribute or menu component data are verified, they may be copied out into working recipe and menu data files. These files may then be loaded at a later date by the data module. The files are named CAA/WRKRECDAT and CAA/WRKMENDAT. The second class of external data files correspond in format to those data files originally received from TSA for the purpose of model development. The file names are CAA/TSARECDAT and CAA/TSAMENDAT. The purpose of the external data files is to provide a source of input that is external to the model itself. These files may consist of specialized data, such as single entree menus for mobilization menu planning, that were originally developed within the model and saved for later use. In addition, they may include data produced by a management information system. The file formats are shown in Chapter 3.

d. Sample Procedures. The user may execute the data module by entering:

RUN CAA/DATAMOD

The user must then respond to the display as shown in Figure 2-3.

```
**WELCOME TO THE ARMY MASTER MENU DATA HANDLING PROGRAM**

**IF NOT FAMILIAR WITH THE PROGRAM STRUCTURE, PLEASE TERMINATE.**

**  THE USER MAY SELECT ANY OF THE FOLLOWING TRANSACTIONS: **

      1    ACCESS THE RECIPE ATTRIBUTE FILE

      2    ACCESS THE MENU COMPONENT FILE

      3    EXECUTE THE PREPROCESSOR

      4    ACCESS THE MENU ATTRIBUTE FILE

      5    GENERATE A RECIPE-MENU CROSS REFERENCE LIST

      6    TERMINATE THIS ROUTINE

**ENTER TRANSACTION NUMBER:
```

Figure 2-3. Data Module Interface

(1) Accessing the Recipe Attribute File. Once the user accesses the recipe attribute file, those actions shown in Figure 2-4 may be taken.

```
**THIS PROGRAM MAINTAINS A DIRECT FILE OF RECIPES AND THEIR  **
**ASSOCIATED ATTRIBUTES. **

**SELECT ONE OF THE FOLLOWING TRANSACTIONS:  **

      1    LIST A PORTION OF THE FILE

      2    LOCATE AN INDIVIDUAL RECIPE

      3    DELETE A RECIPE FROM THE FILE

      4    INSERT A RECIPE INTO THE FILE

      5    MODIFY A RECIPE

      6    LOAD EXTERNAL DATA FILE

      7    TERMINATE THIS ROUTINE

** ENTER TRANSACTION NUMBER:
```

Figure 2-4. Recipe Attribute File Interface

(a) Listing a Portion of the File. A portion of the file may be listed as shown in Figure 2-5. The display is intended to assist the user with input. As an example, "KIND VEG" is to be entered directly below the "AAAA AAA".

```
NOTE:  RECIPES MAY BE LISTED BY KIND OR BY PREFIX.  ENTER KIND OR PREF
FOLLOWED BY A PARAMETER WHICH MAY BE VEG, STA, ENT, DES, OR SAL FOR A
LISTING BY KIND, OR L, Q, ETC. FOR A LISTING BY PREFIX.  EXAMPLE: "KIND
VEG" WILL LIST ALL VEGETABLE RECIPES WHILE "PREF L" WILL LIST ALL RE-
CIPES WITH AN L PREFIX.

ENTER REQUEST:
AAAA AAA
KIND VEG
```

Figure 2-5. Recipe Listing Interface

(b) Locating an Individual Recipe. The procedure for locating data concerning an individual recipe may be displayed by responding to the display shown in Figure 2-6. Recipe numbers should normally correspond to those listed in TM 10-412, although a recipe may be given any identifying number. The data file will be searched until the recipe is found; therefore, if the recipe is not on the data file, there will be a significant delay before the user is told that the recipe cannot be found.

```
THIS ROUTINE WILL LOCATE AND DISPLAY RECIPES FROM THE RECIPE DATA FILE.

HOW MANY RECIPES DO YOU WANT TO SEE?
1

ENTER RECIPE NUMBER:
AAAAAAAAAA
L-150
RECIPE #: L-150 NAME:  CHICKEN CHOW MEIN
KIND:  ENT
FOOD COST: $ 54.98  LABOR: 2.99 MAN-HRS   ACCEPTABILITY: 50.0
NUTRIENTS:
CALORIES:  63418.    PROTEIN:  3866.8     FAT:  2183.6
CALCIUM:   10582.    IRON:  223.4         VITAMIN A: 38148.
THIAMIN:   12.82     RIBOFLAVIN: 61.72    NIACIN: 815.2
VITAMIN C: 2368.2

TRANSACTION COMPLETED.  DO YOU WANT TO DO ANYTHING ELSE WITH THE RECIPE
FILE?

ENTER YES OR NO:
```

Figure 2-6. Procedure for Listing Individual Recipe Data

(c) Deleting a Recipe. The procedure for deleting a data record for a particular recipe is as shown in Figure 2-7. As with locating a recipe, there will be a significant delay if the recipe is not on the data file.

```
THIS ROUTINE WILL DELETE RECIPES FROM THE RECIPE DATA FILE.  HOW MANY
DELETIONS DO YOU WISH TO MAKE?
1
ENTER RECIPE NUMBER:
AAAAAAAAAA
L-150
```

Figure 2-7. Procedure for Deleting a Recipe

(d) Inserting a Recipe. The procedure for inserting data concerning a new recipe with sample entries is shown in Figure 2-8. Care should be taken to ensure that the recipe number has not already been used, since this will cause duplicate records to be entered on the data files. If a variation of an existing recipe is to be entered it may be desirable to give the new recipe the same number with an additional -#. As an example, a variation of L-150 might be L-150-1. Names may not exceed 10 characters in length. All input data must be for 100 servings of the recipe.

```
THIS ROUTINE WILL INSERT RECIPES INTO THE RECIPEINSERT DATA FILE.
HOW MANY ENTRIES DO YOU WISH TO MAKE?

ENTER RECIPE NUMBER AND NAME (NOT TO EXCEED 30 CHARACTERS):
AAAAAAAAAA  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Z-100       HILL OF BEANS
USING A3 FORMAT, ENTER THE KIND OF RECIPE I.E. VEG, DES, ENT :
AAA
VEG
USING F6 FORMAT, ENTER RAW FOOD COST PER 100 SERVINGS:
FFFFFF
25.00
USING F5 FORMAT, ENTER NUMBER OF MAN-HOURS PER 100 SERVINGS:
FFFFF
2.5
USING F5 FORMAT, ENTER ACCEPTABILITY FACTOR:
FFFFF
45
USING F7 FORMAT, ENTER NUTRIENT VALUE PER 100 SERVINGS FOR THE FOLLOWING
FACTORS:
CALORIES:
5000
PROTEIN:
3800
FAT:
2000
CALCIUM:
9000
IRON:
220
VITAMIN A:
35160
THIAMIN:
12
RIBOFLAVIN:
61.7
NIACIN:
820
VITAMIN C:
2300

**RECIPE Z-100 HAS BEEN INSERTED AS FOLLOWS.
RECIPE #:  Z-100     NAME:  HILL OF BEANS      KIND:  VEG
FOOD COST: $ 25.00   LABOR: 2.20 MANHRS  ACCEPTABILITY FACTOR: 45.0%
NUTRIENTS:
CALORIES:      5000. PROTEIN:       3800.0  FAT:       2000.0
CALCIUM:       9000. IRON:           220.0  VITAMIN A: 35160.
THIAMIN:      12.00 RIBOFLAVIN:     61.70  NIACIN:     820.0
VITAMIN C:    2300.0
TRANSACTION COMPLETED.  DO YOU WANT TO DO ANYTHING ELSE WITH THE RECIPE
FILE? ENTER YES OR NO:
```

Figure 2-8. Procedure for Inserting a Recipe

(e) Modifying a Recipe. The procedure for modifying recipe data is shown in Figure 2-9. An example is shown where the cost of serving a "hill of beans" has gone from $25.00 to $30.00.

```
THIS ROUTINE WILL MAKE CHANGES TO CURRENT RECIPES.  ENTER RECIPE NUMBER:
AAAAAAAAAA
Z-100

THE CURRENT DATA FOR RECIPE Z-100  IS AS FOLLOWS:

RECIPE #: Z-100  NAME: HILL OF BEANS   KIND: VEG
FOOD COST: $25.00  LABOR: 2.20 MAN-HRS  ACCEPTABILITY: 45.0
NUTRIENTS:
CALORIES:      5000. PROTEIN:      3800.0 FAT:          2000.0
CALCIUM:       9000. IRON:          220.0  VITAMIN A:    35160.
THIAMIN:       12.00 RIBOFLAVIN:   61.70  NIACIN:       820.0
VITAMIN C:     2300.0

THE FOLLOWING CHANGES MAY BE MADE:

     1    CHANGE RECIPE NAME
     2    CHANGE RECIPE KIND
     3    CHANGE FOOD COST
     4    CHANGE LABOR MAN-HOURS
     5    CHANGE ACCEPTABILITY FACTOR
     6    CHANGE A NUTRITIONAL FACTOR
     7    MAKE NO CHANGES

**ENTER TYPE:
3
ENTER NEW FOOD COST:
FFFFF
30.00

THE CURRENT DATA FOR RECIPE Z-100  IS AS FOLLOWS:

RECIPE #:  Z-100       NAME:  HILL OF BEANS   KIND:  VEG
FOOD COST:  $ 30.00   LABOR:   2.20 MANHRS   ACCEPTABILITY: 45.0
NUTRIENTS:
CALORIES:      5000. PROTEIN:      3800.0 FAT:       2000.0
CALCIUM:       9000. IRON:          220.0 VITAMIN A: 35160.
THIAMIN:       12.00 RIBOFLAVIN:   61.70 NIACIN:     820.0
VITAMIN C:     2300.0

DO YOU WISH TO CHANGE ANY OTHER RECIPES?  ENTER YES OR NO:
```

Figure 2-9. Procedure for Modifying a Recipe

(f) Loading an External Data File. As discussed earlier, an external data file may be loaded. An example of the procedure and the corresponding user display is the user to rapidly load a set of recipe data that may have been current recipe data file. The procedure is intended to allow the user to rapidly load a set of recipe data that may have been developed outside the menu planning model.

```
** CAUTION ** EXECUTION OF THIS ROUTINE WILL DESTROY DATA CURRENTLY EX-
             ISTING ON UNIT 10

**YOU MAY SELECTED ONE OF THE FOLLOWING TYPE TRANSACTIONS:
   1    LOAD A TSA FORMATTED RECIPE DATA FILE
   2    LOAD A WORKING RECIPE DATA FILE WITH THE SAME FORMAT AS THE DI-
             RECT FILE

   3    DO NOT LOAD ANYTHING

**ENTER TYPE:
2
** WAIT. RECIPE FILE IS BEING INITIALIZED FOR DIRECT ACCESS.

   UNCLASSIFIED

   UNCLASSIFIED


** WAIT.  WORKING RECIPE DATA FILE IS BEING LOADED. **

** WAIT.  LOADING CONTINUES.  250 RECORDS LOADED.

** WAIT.  LOADING CONTINUES.  500 RECORDS LOADED.

** WAIT.  LOADING CONTINUES.  750 RECORDS LOADED.

** WAIT.  LOADING CONTINUES. 1000 RECORDS LOADED.

** WAIT.  LOADING CONTINUES. 1250 RECORDS LOADED.

** WAIT.  LOADING CONTINUES. 1500 RECORDS LOADED.


**LOAD COMPLETED. 1674 RECORDS LOADED.


TRANSACTION COMPLETED.  DO YOU WANT TO DO ANYTHING ELSE WITH THE RECIPE
FILE?

ENTER YES OR NO:
```

Figure 2-10. Loading Recipe Data

(g) Terminating the Routine. Access to the recipe attribute file may be terminated by entering the appropriate transaction number. This does not terminate access to the data module. The user may then continue working with the data module by responding to the display shown earlier in Figure 2-3.

(2) Accessing the Menu Component File. Once the user accesses the menu component file, those actions shown in Figure 2-11 may be taken.

```
**THIS PROGRAM MAINTAINS A DIRECT FILE OF MENUS AND THEIR  **
**ASSOCIATED RECIPES.  **

**SELECT ONE OF THE FOLLOWING TRANSACTIONS:  **

     1    LIST A PORTION OF THE FILE

     2    LOCATE AN INDIVIDUAL MENU

     3    DELETE A MENU FROM THE FILE

     4    INSERT A MENU INTO THE FILE

     5    MODIFY A MENU

     6    LOAD EXTERNAL DATA FILE

     7    TERMINATE THIS ROUTINE

**ENTER TRANSACTION NUMBER:
```

Figure 2-11. Menu Component File Interface

(a) Listing a Portion of the File. A portion of the menu component file may be listed as shown in Figure 2-12. Listings are automatically queued to the printer.

```
NOTE:  MENUS MAY BE LISTED BY MEAL.
     1    BREAKFAST
     2    LUNCH
     3    SHORT ORDER
     4    DINNER

ENTER MEAL TYPE:
1
3

**WAIT.  MENU FILE IS BEING LISTED.**

TRANSACTION COMPLETED.  DO YOU WANT TO DO ANYTHING ELSE WITH THE MENU
FILE?

ENTER YES OR NO:
```

Figure 2-12. Menu Component Listing Interface

(b) Locating an Individual Menu. Data concerning a particular menu may be accessed by responding to the display shown in Figure 2-13. Menu numbers are normally assigned by the model at loading time. The format for menu numbers is A-###, where A is either B, S, L, or D for breakfast, short order, lunch or dinner ### is a three-digit number identifying the particular menu. These numbers are assigned sequentially so that if, as an example, 37 short order menus are loaded, they will be numbered S-001 through S-037. Numbers less than three digits should be preceded by zeros, i.e., B-002, and not B-2.

```
THIS ROUTINE WILL LOCATE AND DISPLAY MENUS FROM THE MENU DATA FILE.

ENTER MENU NUMBER:
AAAAAAAAAA
L-098
**MENU L-098      CONSISTS OF THE FOLLOWING 18 RECIPES:

                  C-12      C-5       D-16      I-48
                  J-8-2     L-46-1    L-88-2    M-44
                  M-71      P-25      Q-17-1    Q-57
                  X-141     X-48      X-49      X-50
                  X-51      X-71

DO YOU WISH TO SEE ANY OTHER MENUS?
ENTER YES OR NO:
```

Figure 2-13. Procedure for Listing Individual Menu Component Data

(c) Inserting a Menu. The procedure for inserting data concerning a menu is shown in Figure 2-14. The menu numbering convention should be observed and care should be taken to ensure that a previously used menu number is not used.

```
THIS ROUTINE WILL INSERT MENUS INTO THE MENU DATA FILE.  HOW MANY EN-
TRIES DO YOU WISH TO MAKE?
1
ENTER MENU NUMBER:
AAAAAAAAAA
S-100
ENTER THE NUMBER OF RECIPES IN MENU S-100:
NOTE:  THERE CAN BE NO MORE THAN 30 RECIPES.
II
10
ENTER RECIPE  1 OF 10 RECIPES:
A-1
ENTER RECIPE  2 OF 10 RECIPES:
A-2
ENTER RECIPE  3 OF 10 RECIPES:
A-3
ENTER RECIPE  4 OF 10 RECIPES:
A-4
ENTER RECIPE  5 OF 10 RECIPES:
A-5
ENTER RECIPE  6 OF 10 RECIPES:
A-6
ENTER RECIPE  7 OF 10 RECIPES:
A-7
ENTER RECIPE  8 OF 10 RECIPES:
A-8
ENTER RECIPE  9 OF 10 RECIPES:
A-9
ENTER RECIPE 10 OF 10 RECIPES:
A-10

**MENU S-100      HAS BEEN INSERTED AS FOLLOWS:
MENU #:S-100      RECIPES:  A-1      A-2      A-3      A-4
                            A-5      A-6      A-7      A-8
                            A-9      A-10

TRANSACTION COMPLETED.  DO YOU WANT TO DO ANYTHING ELSE WITH THE MENU
FILE?
ENTER YES OR NO:
```

Figure 2-14. Procedure for Inserting Menu Component Data

(d) Modifying Menu Component Data. The procedure for modifying menu component data is shown in Figure 2-15. In this example, recipe A-5 is replaced by recipe X-15.

```
THIS ROUTINE WILL MAKE CHANGES TO CURRENT MENUS.
ENTER MENU NUMBER:
AAAAAAAAAA
S-100
THE CURRENT DATA FOR MENU S-100     IS AS FOLLOWS:
MENU #:S-100      RECIPES:  A-1      A-2      A-3      A-4
                            A-5      A-6      A-7      A-8
                            A-9      A-10

THE FOLLOWING CHANGES MAY BE MADE:

     1    CHANGE A RECIPE NUMBER

     2    DELETE A RECIPE

     3    ADD A RECIPE

     4    MAKE NO CHANGES

**ENTER TYPE:
1
HOW MANY RECIPES DO YOU WANT TO REPLACE?
1
ENTER OLD RECIPE NUMBER FOLLOWED BY NEW RECIPE NUMBER
AAAAAAAAAA  AAAAAAAAAA
A-5         X-15

THE CURRENT DATA FOR MENU S-100        IS AS FOLLOWS:
MENU #:S-100      RECIPES:  A-1      A-2      A-3      X-4
                            X-15     A-6      A-7      A-8
                            A-9      A-10

DO YOU WISH TO CHANGE ANY OTHER MENUS?
ENTER YES OR NO:
```

Figure 2-15. Procedure for Modifying Menu Component Data

(e) Deleting Menu Component Data. The procedure for removing a menu from the menu component data file is shown in Figure 2-15.

```
THIS ROUTINE WILL DELETE MENUS FROM THE MENU DATA FILE.  HOW MANY DELE-
TIONS DO YOU WISH TO MAKE?
1
ENTER MENU NUMBER:
AAAAAAAAAA
S-100
MENU NUMBER S-100 HAS BEEN DELETED FROM THE FILE.

TRANSACTION COMPLETED.  DO YOU WANT TO DO ANYTHING ELSE WITH THE MENU
FILE?

ENTER YES OR NO:
```

Figure 2-16. Procedure for Deleting Menu Component Data

(f) Loading External Data Files. As with the recipe attribute file, menu component data may also be loaded. The procedure for loading an external data file is shown in Figure 2-17.

```
** CAUTION ** EXECUTION OF THIS ROUTINE WILL DESTROY DATA CURRENTLY
              EXISTING OF UNIT 12

YOU MAY SELECT ONE OF THE FOLLOWING TYPE TRANSACTIONS:

   1     LOAD A TSA FORMATTED MENU DATA FILE

   2     LOAD A WORKING MENU DATA FILE WITH THE SAME FORMAT AS THE
               DIRECT FILE

   3     DO NOT LOAD ANYTHING

** WAIT.  MENU FILE IS BEING INITIALIZED FOR DIRECT ACCESS.

** WAIT.  WORKING MENU DATA FILE IS BEING LOADED.  **

** WAIT.  LOADING CONTINUES.  250 RECORDS LOADED.


**LOAD COMPLETED.  325 RECORDS LOADED.
66 BREAKFAST MENUS
112 LUNCH MENUS
37 SHORT ORDER MENUS
110 DINNER MENUS


TRANSACTION COMPLETED.  DO YOU WANT TO DO ANYTHING ELSE WITH THE MENU
FILE?

ENTER YES OR NO:
```

Figure 2-17. Procedure for Loading Menu Component Data

(g) Terminating Access to the Menu Component File. Access to the menu component data file may be accessed by responding with the appropriate answer or by entering a "7" when presented with the display shown earlier in Figure 2-11. The user may then continue working with the data module by responding to the display that was shown in Figure 2-3.

(3) Executing the Preprocessor. The preprocessor performs the task of identifying the recipes that are in the menu component file in terms of their attributes. Data for each recipe in each menu is retrieved; therefore, if a menu is to include a particular recipe, data for that recipe must be in the recipe attribute file. (Whenever a recipe cannot be found, that recipe number will be displayed and the routine will terminate.) Data generated by the preprocessor are entered into the menu attribute file. An example of executing the preprocessor is shown in Figure 2-18.

```
**  WELCOME TO THE ARMY MASTER MENU DATA HANDLING PROGRAM    **

**  IF NOT FAMILIAR WITH THE PROGRAM STRUCTURE, PLEASE TERMINATE.  **

**    THE USER MAY SELECT ANY OF THE FOLLOWING TRANSACTIONS:
**

        1    ACCESS THE RECIPE ATTRIBUTE FILE

        2    ACCESS THE MENU COMPONENT FILE

        3    EXECUTE THE PREPROCESSOR

        4    ACCESS THE MENU ATTRIBUTE FILE

        5    GENERATE A RECIPE-MENU CROSS REFERENCE LIST

        6    TERMINATE THIS ROUTINE

**  ENTER TRANSACTION NUMBER:

3

**WAIT.  THE MENU ATTRIBUTE FILE IS BEING GENERATED.

**WAIT.  50 MENUS HAVE BEEN PROCESSED.**

**WAIT.  100 MENUS HAVE BEEN PROCESSED.**

**WAIT.  150 MENUS HAVE BEEN PROCESSED.**

**WAIT.  200 MENUS HAVE BEEN PROCESSED.**

**WAIT.  250 MENUS HAVE BEEN PROCESSED.**

**WAIT.  300 MENUS HAVE BEEN PROCESSED.**

**MENU ATTRIBUTE FILE COMPLETED ON UNIT 14.  325 MENUS PROCESSED.**
```

Figure 2-18. Executing the Preprocessor

(4) Accessing the Menu Attribute File. Unlike the recipe attribute file, or the menu component file, the data in the menu attribute file is not to be changed. The user is allowed to display selected portions of the menu attribute data file. The options available to the user with sample output are shown in Figure 2-19. Except for data concerning an individual menu, listings are sent to the printer. If the user desires information concerning the recipes comprising the selected menu, that listing is also sent to the printer. Menu attribute data is related to serving 100 persons.

```
*  THIS ROUTINE PRODUCES A LISTING     *
*  OF THE CURRENT MENU ATTRIBUTE DATA. *

**  SELECT ONE OF THE FOLLOWING TRANSACTIONS:

     1    LIST ALL THE MENUS

     2    LIST BREAKFAST MENUS

     3    LIST LUNCH MENUS

     4    LIST DINNER MENUS

     5    LIST SHORT ORDER MENUS

     6    LIST AN INDIVIDUAL MENU

     7    PRODUCE NO LISTING

**ENTER TRANSACTION TYPE:
6
**ENTER MENU NUMBER:
AAAAAAAAAA
L-098
MENU NO. = L-098
      ACCEPTABILITY    FOOD COST    LABOR MANHOURS
          65.66           93.55         26.20
      CALORIES     PROTEIN      FAT          CALCIUM        IRON
      151400.83     4576.92     7525.55      61530.15      576.83
      VITAMIN A    THIAMIN   RIBOFLAVIN     NIACIN       VITAMIN C
      608097.72     64.08       107.29      714.41        5186.92

**DO YOU WANT A LISTING OF THE RECIPE ATTRIBUTE DATA FOR EACH OF THE RE-
CIPES IN MENU L-098?
**ENTER YES OR NO:
YES
**LISTING COMPLETED.     1 MENU LISTED
```

Figure 2-19. Accessing the Menu Attribute File

(5) Generating a Recipe-menu Cross Reference List. Although the menu component file provides information about which recipes are included in each menu, it is also necessary to know in which menus certain recipes appear. This information may be especially important to the user before executing the preprocessor. If the preprocessor fails, the user is then able to locate all the menus in which the recipe of concern is included. The procedure for generating a recipe-menu cross reference list is dependent on the ability to sort various files. At the time the model was first implemented, a FORTRAN callable system sort routine was not available; therefore, a natural merge sort routine was developed for inclusion in the model. This sort routine was somewhat slower than might be desirable, and therefore an alternative approach was developed which was not only much faster, but also allowed the user to continue operating at the terminal while the cross reference list was being generated. The two different procedures are described below. The cross reference list will be queued to the printer.

(a) The procedure for generating a recipe-menu cross reference list within the data module is shown in Figure 2-20. This procedure may be fairly slow.

```
**  WELCOME TO THE ARMY MASTER MENU DATA HANDLING PROGRAM    **

**  IF NOT FAMILIAR WITH THE PROGRAM STRUCTURE, PLEASE TERMINATE.  **

**    THE USER MAY SELECT ANY OF THE FOLLOWING TRANSACTIONS:


        1    ACCESS THE RECIPE ATTRIBUTE FILE

        2    ACCESS THE MENU COMPONENT FILE

        3    EXECUTE THE PREPROCESSOR

        4    ACCESS THE MENU ATTRIBUTE FILE

        5    GENERATE A RECIPE-MENU CROSS REFERENCE LIST

        6    TERMINATE THIS ROUTINE

**  ENTER TRANSACTION NUMBER:

5

**WAIT.  CROSS-REFERENCE IS BEING LISTED.

WAIT.  CROSS-REFERENCE LIST IS BEING SORTED.

**CROSS-REFERENCE LIST COMPLETED.
```

Figure 2-20. Generating a Recipe-Menu Cross Reference Listing

(b) An alternate procedure for generating a recipe-menu cross reference list outside the data module is to simply enter the following command after terminating access to the data handling module:

"START CAA/FILEXREF/JOB"

This command generates a separate job which will send the cross-reference listing to the printer.

(6) Terminating Access to the Data Handling Module. Access to the data module may be terminated by responding appropriately to the display shown earlier in Figure 2-3. Printouts that may have been generated during the session may be queued to the printer by entering either a "SPLIT" command or a "BYE" command.

e. As with any system, the information that goes into the menu planning model is of the utmost importance. Without reliable data, there is no sense in continuing with the menu planning process. The data module, however is intended to make the maintenance of data a reliable process leading to a consistent analysis of the relative worth of the various menus that may ultimately be served to the soldier in the field. Once the user is satisfied with the data contained in the menu attribute file, it is possible to proceed to the parameterization module where the menu planning parameters may be established.

2-4. PARAMETERIZATION MODULE

a. Purpose. The parameterization module allows the user to establish the menu planning parameters. The module is represented in Figure 2-21. As can be seen by referring back to Figure 2-1, the parameterization module is the link between the data module and the solution module.

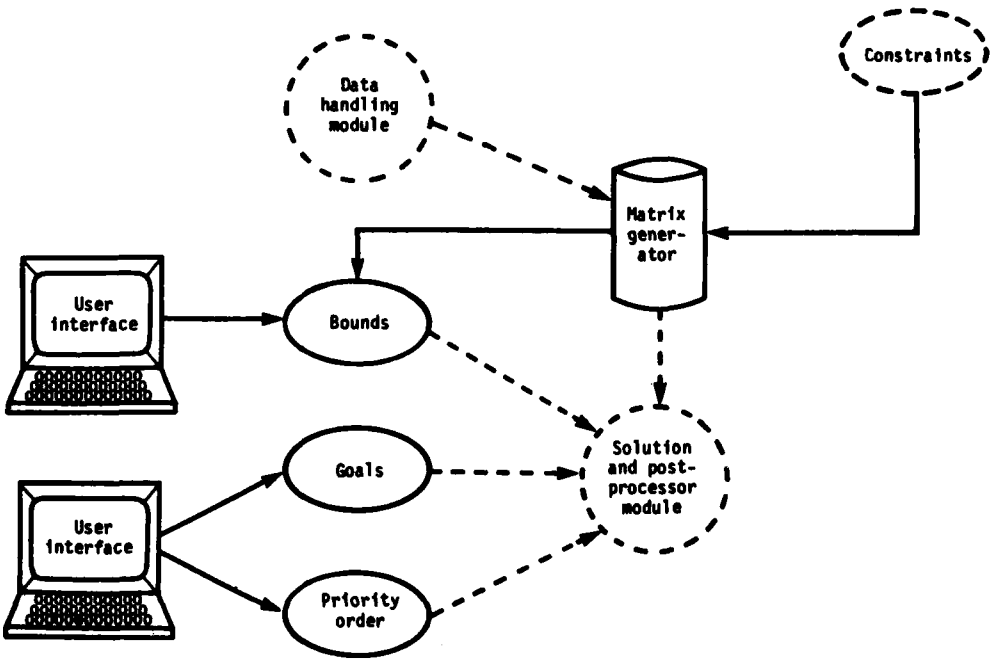


Figure 2-21. Parameterization Module

b. Data Sources. The link between the parameterization and the data handling module is through the menu attribute data file. The validity of the data in that file should be verified before establishing menu planning parameters. All other data files are created and maintained within the parameterization module as explained below.

c. Input Requirements. As explained above, the menu attribute file is the only external file required as input to the parameterization module. Several other files are maintained by the module and are accessed through user interfaces.

(1) Menu Attribute File. See paragraph 2-2c(3).

(2) Goal Programing Problem Matrix. As will be explained later, this file is created and maintained by the parameterization module. The format of the file corresponds to a standard input format for many large scale LP packages. The file name is CAA/GPDATA.

(3) Bounds File. This file is also created within the parameterization module. The file name is CAA/BOUNDS.

(4) Goals File. This file is created and maintained by the parameterization module. In goal programing terminology, the goals are often referred to as the right hand side or RHS; accordingly, the file name is CAA/RHS.

(5) Priority File. The priority file is created and maintained by the parameterization module. The file name is CAA/PRIORS.

d. Sample Procedures. The user may execute the parameterization module by entering:

RUN CAA/PARAMOD

The user must then respond to the display as shown in Figure 2-22.

```
**  WELCOME TO THE MASTER MENU PARAMETERIZATION PROGRAM **


**  IF NOT FAMILIAR WITH THE PROGRAM STRUCTURE, PLEASE TERMINATE.**
**  THE USER MAY SELECT ANY OF THE FOLLOWING TRANSACTIONS:  **

     1    EXECUTE THE MATRIX GENERATOR

     2    ACCESS THE BOUNDS FILE

     3    ACCESS THE GOALS FILE

     4    ACCESS THE PRIORITY ORDERING

     5    TERMINATE THIS ROUTINE

**  ENTER TRANSACTION NUMBER:
```

Figure 2-22. Parameterization Module Interface

(1) Executing the Matrix Generator. The matrix generator must be executed in order to produce a current GP problem matrix from which the solution will be derived. As shown in Figure 2-23, two actions take place when the problem matrix is generated. One is the creation of the problem matrix itself, and the other is the creation of the initial bounds. The initial bounds are upper limits on the number of times that a menu of any meal type may be repeated during the menu planning cycle.

```
* AN UPPER BOUND SHOULD BE PLACED ON EACH MEAL.  THIS UPPER BOUND SHOULD
BE THE LARGEST NUMBER OF TIMES THAT ANY MENU IS TO BE REPEATED DURING
THE MENU CYCLE.  AS AN EXAMPLE:  IF BREAKFAST IS GIVEN AN UPPER BOUND OF
4 THEN NO BREAKFAST MENU WILL BE REPEATED IN ITS ENTIRETY MORE THAN 4
TIMES DURING THE CYCLE.  AN UPPER BOUND OF 1 ON A MEAL MEANS THAT NO
MENU WILL BE REPEATED FOR THAT MEAL.

* WARNING:  A SITUATION SUCH AS PLACING AN UPPER BOUND OF 1 ON A MEAL
FOR A 365 DAY CYCLE WHEN THERE ARE ONLY 100 MENUS OF THAT MEAL TO SELECT
FROM IS INFEASIBLE.

* PLEASE ENTER UPPER LIMITS ON MENUS FOR BREAKFAST, SHORT ORDER, LUNCH,
AND DINNER.

4 2 2 2

MENU UPPER LIMITS ARE:
        BREAKFAST               4.00
        SHORT ORDER             2.00
        LUNCH                   2.00
        DINNER                  2.00

GENERATION OF THE PROBLEM MATRIX IS COMPLETE.
```

Figure 2-23. Executing the Matrix Generator

(2) Accessing the Bounds File. Although upper bounds are initially set in by the matrix generator, bounds on particular menus may be established by accessing the bounds file. There are three types of bounds, only one of which may be applied at a time: upper (UP), lower (LO), and fixed (FX). An upper bound is the maximum number of times that a menu may be repeated during the cycle. A lower bound is the least number of times that a menu is to be served during a cycle. A fixed bound requires that a menu is to be served exactly that many times during a cycle. As an example, it may be desirable to set a fixed bound of 1 on a Thanksgiving dinner menu if planning for the month of November, or it may be necessary to set a fixed bound of 0 on a meal that for one reason or another is not to be included in the cycle. The procedure for accessing the bounds file along with an example is shown in Figure 2-24.

```
                    ***** MENU BOUNDS *****

EACH MENU WAS GIVEN AN UPPER BOUND BY THE MATRIX GENERATOR.

THIS ROUTINE ALLOW THE USER TO REVISE THESE BOUNDS TO EITHER A NEW UPPER
BOUND, A LOWER BOUND OR A FIXED VALUE.

HOW MANY BOUNDS DO YOU WANT TO REVISE?
1
** ENTER MENU NUMBER:
L-050
**CURRENT BOUND FOR MENU L-050 IS:  UP            2.

ENTER NEW BOUND IN THE FOLLOWING FORMAT:

AA FFFFFFFFFFF
FX 0.
** NEW BOUND FOR MENU L-050 IS:    FX           0.
** 1 REVISION COMPLETED.
```

Figure 2-24. Accessing the Bounds File

(3) Accessing the Goals File. The menu planning goals may be accessed through the parameterization module by responding to the display shown earlier in Figure 2-22. A sample procedure is shown in Figure 2-25.

```
                    *** CURRENT MENU PLANNING GOALS ***

LENGTH OF MENU PLANNING CYCLE: 42.0 DAYS

BASIC DAILY FOOD ALLOWANCE:  $ 3.47

                ACCEPTABILITY(%)      LABOR (MAN-HOURS/MEAL)

BREAKFAST:          99.                    14.
SHORT ORDER:        69.                    12.
LUNCH:              73.                    16.
DINNER:             79.                    16.

*NUTRITION*

CALORIES:          3200.00      VITAMIN A:          5000.00
PROTEIN:            100.00      THIAMIN:               1.60
FAT:                   .00      RIBOFLAVIN:            2.00
CALCIUM:            800.00      NIACIN:               21.00
IRON:                18.00      VITAMIN C:            60.00

**DO YOU WANT TO CHANGE ANY OF THE ABOVE GOALS? ANSWER YES OR NO.
YES
**THE FOLLOWING TYPE GOALS MAY BE CHANGED:

     1    LENGTH OF MENU PLANNING CYCLE

     2    ACCEPTABILITY

     3    FOOD COST

     4    LABOR

     5    NUTRITION

**ENTER TYPE:
3
CURRENT BASIC DAILY FOOD ALLOWANCE IS $  3.47

ENTER NEW VALUE:
3.86
*** CURRENT MENU PLANNING GOALS ***

LENGTH OF MENU PLANNING CYCLE: 42.0 DAYS

BASIC DAILY FOOD ALLOWANCE:  $ 3.86

                ACCEPTABILITY(%)      LABOR (MANHOURS/MEAL)

   BREAKFAST:          99.                    14.
   SHORT ORDER:        69.                    12.
   LUNCH:              73.                    16.
   DINNER:             79.                    16.

*NUTRITION*

CALORIES:          3200.00      VITAMIN A:          5000.00
PROTEIN:            100.00      THIAMIN:               1.60
FAT:                   .00      RIBOFLAVIN:            2.00
CALCIUM:            800.00      NIACIN:               21.00
IRON:                18.00      VITAMIN C:            60.00

**DO YOU WANT TO CHANGE ANY OF THE ABOVE GOALS?  ANSWER YES OR NO.
```

Figure 2-25. Accessing the Goals File

(4) Accessing the Priority File. The priority file may be accessed by responding to the display shown earlier in figure 2-22. The purpose of accessing the priority file is to establish the order in which the menu attributes are to be considered in the design of the subsequently produced menu plan. A sample procedure is shown in Figure 2-26.

```
                    ***** PRIORITY ORDER *****

SELECT THE PRIORITY ORDERING BY ENTERING A 1, 2, 3, OR 4 AFTER EACH
ATTRIBUTE AS IT IS DISPLAYED:

NUTRITION:
2
ACCEPTABILITY:
3
FOOD COST:
4
LABOR COST:
1

THANK YOU.

THE FOUR MENU ATTRIBUTES WILL BE ORDERED IN THE FOLLOWING PRIORITIES:

     1    LABOR COST:

     2    NUTRITION:

     3    ACCEPTABILITY:

     4    FOOD COST:

ARE THERE ANY CHANGES?
```

Figure 2-26. Accessing the Priority File

(5) Terminating Access to the Parameterization Module. Access to the parameterization module may be terminated by responding appropriately to the display shown earlier in Figure 2-22. No printouts are generated by the parameterization module and therefore it is not necessary to enter the "SPLIT" command as in the data module.

e. It should be remembered that the parameterization module is the link between the data module and the solution module. The problem matrix is based on the data contained in the menu attribute file, and the solution to be generated in the solution module is produced strictly within the framework of the parameters established in the parameterization module.

2-5. SOLUTION MODULE

a. Purpose. The solution module allows the user to produce a menu plan based on the previously entered data and within the established parameters. The module is represented in Figure 2-27. As can be seen by referring back to Figure 2-1, the solution module is the final step in the menu planning process.

b. Data Sources. All data used by the solution module are developed in either the data module or the parameterization module. In the strictest sense, those data files produced by the parameterization module are sufficient for execution of the solution module, however some of the reports produced by the postprocessor are dependent upon the data files maintained by the data module.

c. Input Requirements

(1) Recipe attribute file. See paragraph 2-2c(1).

(2) Menu component file. See paragraph 2-2c(2).

(3) Menu attribute file. See paragraph 2-2c(3).

(4) GP problem matrix file. See paragraph 2-3c(2).

(5) Bounds file. See paragraph 2-3c(3).

(6) Goals file. See paragraph 2-3c(4).

(7) Priority file. See paragraph 2-3c(5).

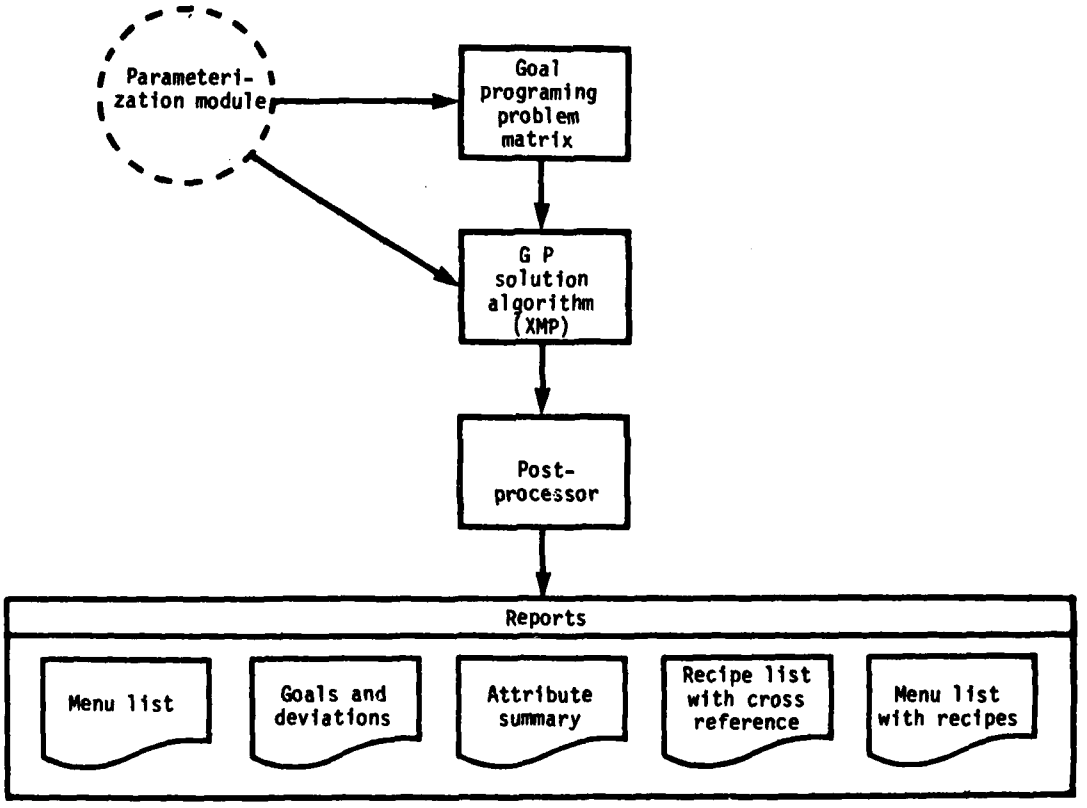


Figure 2-27. Solution Module

d. Sample Procedures. The user may execute the solution module by entering:

START CAA/MENUPLAN

After entering the above command, the user may sign off the terminal or go on to other work. Care should be taken that none of the input files are changed until the menu plan is completed. A printout of the five reports shown in Figure 2-27 will be produced and sent to the printer.

e. As mentioned in paragraph 2-2d(5), a FORTRAN callable system sort was not available at the time the model was first implemented, and therefore an alternate procedure for producing a cross reference listing of the solution is to simply enter the following command:

"START CAA/SOLNXREF/JOB"

This command generates a separate recipe-menu cross reference listing that will be queued to the printer.

2-6. SAMPLE SOLUTION. The following discussion is intended to provide the user with an overview of the procedures for generating a menu plan.

a. General Solution Procedure. Since the generation of a menu plan is the final step in the process, the steps that precede it impact on the validity of the solution. The general sequence of actions taken in producing a menu plan are as follows:

- Verify validity of data set
- Generate preprocessor to produce the menu attribute file
- Execute matrix generator
- Establish bounds as necessary
- Verify goals
- Enter priority order
- Execute solution module
- Examine solution

b. Sample Solution. As always the validity of the data set is of the utmost importance, however the assumption in this discussion is that the data set has been validated and that the menu attribute file has been generated successfully.

(1) Upper Bounds. As explained earlier, the matrix generator not only generates the problem matrix, but also sets in the initial upper bounds. The upper bounds shown in Figure 2-23 are applied to this example also. This means that no breakfast menu may be repeated more than 4 times during the cycle, while no lunch, dinner or short order menu may be repeated more than twice.

(2) Goals. The goals to be used in this particular example are shown in Figure 2-28. These goals tell the user that a 42 day menu plan is to be generated. The basic daily food allowance for the period is $3.47, and the nutritional standards of AR 30-1 are to be applied. The acceptability and labor goals correspond to very high goals for those two attributes. The menu planner may wish to change them to more appropriate goals, using the parameterizaton module, depending on the solution.

*** CURRENT MENU PLANNING GOALS ***

LENGTH OF MENU PLANNING CYCLE: 42.0 DAYS

BASIC DAILY FOOD ALLOWANCE: $ 3.47

	ACCEPTABILITY(%)	LABOR (MANHOURS/MEAL)
BREAKFAST:	99.	14.
SHORT ORDER:	69.	12.
LUNCH:	73.	16.
DINNER:	79.	16.

NUTRITION

CALORIES:	3200.00	VITAMIN A:	5000.00
PROTEIN:	100.00	THIAMIN:	1.60
FAT:	.00	RIBOFLAVIN:	2.00
CALCIUM:	800.00	NIACIN:	21.00
IRON:	18.00	VITAMIN C:	60.00

Figure 2-28. Sample Goals

(3) Priority Order. The order in which the four attributes are prioritized may change depending on the particular situation at the time the plan is being generated. The priority order may also be revised if the solution is not satisfactory. In this particular example the attributes are to be prioritized in the following order: Acceptability, Nutrition, Food Cost, and Labor Cost. This is shown in Figure 2-29.

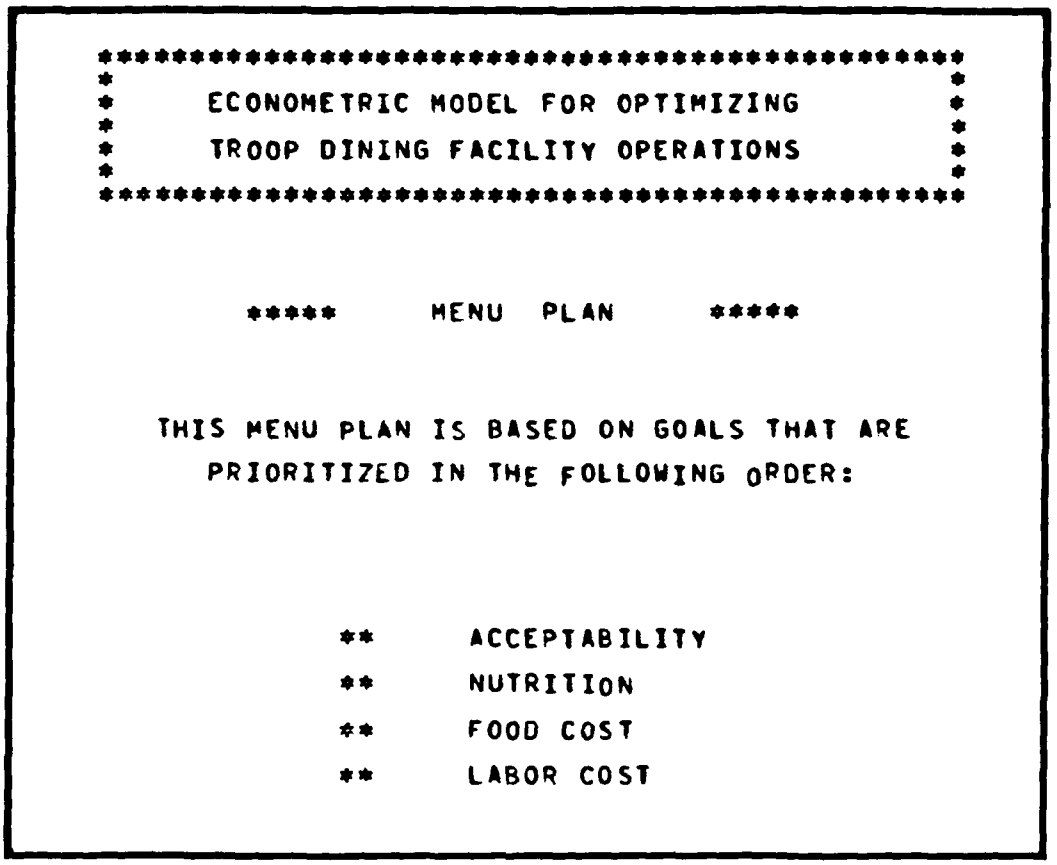


Figure 2-29. Priority Order

(4) Analysis of Solution. Once the parameters have been established as explained, the solution may be generated. The actual creation of a menu plan may be an iterative process because an analysis of the solution may reveal various factors that are unsatisfactory for one reason or another. As an example, goal achievement may be unsatisfactory, or a particular recipe may be served too often during the cycle. Each solution results in the creation of five reports which should be examined closely.

(a) Menu List. The menu list for the current example is shown in Figure 2-30. This simply lists the selected menus and the number of times each is to be served during the cycle.

BREAKFAST		LUNCH		SHORT-ORDER		DINNER	
MENU SELECTED	NUMBER OF SERVING	MENU SELECTED	NUMBER OF SERVING	MENU SELECTED	NUMBER OF SERVING	MENU SELECTED	NUMBER OF SERVING
B-003	#	L-002	2	S-001	2	D-001	2
B-006	#	L-005	2	S-002	2	D-003	2
B-012	#	L-014	2	S-003	2	D-004	2
B-014	#	L-015	2	S-004	2	D-019	2
B-018	#	L-020	2	S-007	2	D-030	2
B-022	#	L-023	2	S-009	2	D-048	2
B-035	#	L-035	2	S-011	2	D-052	2
B-041	#	L-036	2	S-012	2	D-054	2
B-050	#	L-043	2	S-016	2	D-057	2
B-051	#	L-050	2	S-017	2	D-058	2
B-063	2	L-053	2	S-018	2	D-063	2
		L-059	2	S-023	2	D-069	2
		L-063	2	S-024	2	D-072	2
		L-065	2	S-025	2	D-077	2
		L-074	2	S-027	2	D-078	2
		L-079	2	S-028	2	D-079	2
		L-093	2	S-031	2	D-090	2
		L-099	2	S-032	2	D-093	2
		L-106	2	S-033	2	D-095	2
		L-110	2	S-035	2	D-100	2
		L-111	2	S-037	2	D-105	2
11		21		21		21	

Figure 2-30. Sample Menu List

(b) Attribute Summary. The summary of the menu plan in terms of the attributes is shown in Figure 2-31. Included is information concerning the average daily values of each of the attributes for combinations of three meals per day: breakfast-lunch-dinner (B-L-D), and breakfast-short order-dinner (B-S-D). At the bottom are indicated the number of menus of each meal type selected. It is possible to have a number that is not in keeping with the cycle length, i.e., 41 breakfast menus for a 42-day cycle. In this case, bounds should be adjusted and the solution regenerated.

(c) Goals and Deviations. This report is one of the most helpful in a quick analysis of the solution. The information provided in this report and shown in Figure 2-32 tells the user how well this solution satisfies the various goals.

(5) Menus with Associated Recipes. Because the menu list simply displays the selected menus, it may also be desirable to see a listing of the actual composition of those menus in terms of recipes, and therefore, the report shown in Figure 2-33 is provided. Also provided is information concerning the number of times that each menu is to be served during the cycle.

(6) Recipe-menu Cross Reference List. The recipe-menu cross reference list tells the menu planner how many times individual recipes are to be served in the plan and in which menus those recipes are included. This information allows the menu planner to make decisions concerning the frequency with which various recipes should be served. In those cases when a recipe is being served too often, the menu planner may find it desirable to exclude some of the menus in which that recipe appears from the solution by adjusting the appropriate bounds. A portion of the recipe-menu cross reference list for this example is shown at Figure 2-34.

	BREAKFAST AVG	BREAKFAST MIN	BREAKFAST MAX	SHORT ORDER AVG	SHORT ORDER MIN	SHORT ORDER MAX	LUNCH AVG	LUNCH MIN	LUNCH MAX	DINNER AVG	DINNER MIN	DINNER MAX	DAILY B-L-D	DAILY B-S-D
FOOD COST ($/IND	.55	.48	.62	.63	.54	.74	1.30	.93	2.43	1.77	1.04	2.52	3.62	2.96
LABOR (MAN-HRS):	21.	20.	25.	11.	10.	12.	27.	21.	34.	26.	19.	35.	74.	59.
ACCEPTABILITY:	99.	98.	99.	65.	62.	69.	70.	68.	73.	74.	71.	79.		
**NUTRIENTS:														
CALORIES:	1194.6	962.2	1528.3	1212.9	1033.7	1396.6	1571.2	1181.0	2308.0	1616.0	1240.0	2308.0	4381.8	4023.5
PROTEIN (GM)	46.6	38.3	73.8	74.6	59.6	87.9	57.1	37.6	83.2	60.5	42.9	83.2	164.3	181.7
FAT (GM)	54.0	46.5	66.8	57.3	48.2	69.5	68.2	49.8	97.4	74.8	44.3	97.4	197.1	186.2
CALCIUM (MG)	730.8	660.3	855.2	404.5	345.5	562.8	690.3	609.7	774.8	666.9	591.1	795.7	2088.0	1802.3
IRON (MG)	8.2	5.2	18.3	20.7	16.8	28.7	10.0	6.3	28.6	8.8	6.4	12.5	27.0	37.6
VITAMIN A (IU)	2056.	1720.	2077.	707.	477.	1225.	4933.	1436.	10810.	3716.	1436.	8129.	10705.	6479.
THIAMIN (MG)	1.15	.48	3.43	3.55	2.97	4.19	1.01	.64	1.49	.89	.61	1.49	3.05	5.59
RIBOFLAVIN (MG)	1.39	1.15	2.11	4.73	4.26	5.14	1.86	1.18	3.07	1.60	1.15	2.96	4.85	7.71
NIACIN (MG)	6.27	4.14	9.83	30.84	24.81	35.83	11.84	8.39	22.17	11.88	9.04	22.17	29.99	48.99
VITAMIN C (MG)	18.50	8.68	50.13	47.86	45.85	55.71	78.84	38.22	119.35	64.40	26.08	111.48	161.73	130.75
NUMBER OF MENUS:		42			42			42			42			

Figure 2-31. Sample Menu Attribute Summary

ITEM	GOAL	DEVIATION	ACHIEVEMENT
ACCEPTABILITY (%)			
BREAKFAST	99.0	-.4	98.6
SHORT-ORDER	69.0	-4.0	65.0
LUNCH	73.0	-2.6	70.4
DINNER	79.0	-5.2	73.8
FOOD COST ($/INDIVIDUAL)			
BREAKFAST	.69	-.14	.55
SHORT-ORDER	.83	-.45	.38
LUNCH	.56	-.03	.52
DINNER	1.39	.38	1.77
LABOR (MAN-HOUR/MEAL)			
BREAKFAST	14.0	7.4	21.4
SHORT-ORDER	12.0	-.6	11.4
LUNCH	16.0	11.5	27.5
DINNER	16.0	9.9	25.9
FAT-CAL RATIO (INDIV/DAY)			
BREAKFAST	.00	8.61	8.61
SHORT-ORDER	.00	30.90	30.90
LUNCH	.00	-14.19	-14.19
DINNER	.00	26.49	26.49
NUTRIENTS (INDIV/DAY)			
CALORIES	3200.00	966.80	4166.80
PROTEIN (GM)	100.00	74.73	174.73
FAT	.00	190.52	190.52
CALCIUM (MG)	800.00	1116.57	1916.57
IRON (MG)	18.00	15.39	33.39
VITAMIN A (IU)	5000.00	3169.26	8169.26
THIAMIN (MG)	1.60	2.98	4.58
RIBOFLAVIN (MG)	2.00	4.57	6.57
NIACIN (MG)	21.00	20.39	41.39
VITAMIN C (MG)	60.00	83.15	143.15

Figure 2-32. Goals and Deviations

```
**MENU NUMBER:  B-003     IS SERVED     4 TIMES.**
   RECIPE:  C-12       ,HOT TEA                          ,KIND :  OTH
   RECIPE:  C-5        ,COFFEE AUTOMATIC URN             ,KIND :  OTH
   RECIPE:  D-22       ,FRENCH TOAST                     ,KIND :  ENT
   RECIPE:  D-29-1     ,BLUEBERRY MUFFINS                ,KIND :  OTH
   RECIPE:  F-10       ,GRIDDLE FRIED EGGS               ,KIND :  ENT
   RECIPE:  F-11-2     ,CHEESE OMELETTE                  ,KIND :  ENT
   RECIPE:  F-13       ,SCRAMBLED EGGS                   ,KIND :  ENT
   RECIPE:  L-2        ,OVEN FRIED BACON                 ,KIND :  ENT
   RECIPE:  L-88       ,GRILLED SAUSAGE LINKS            ,KIND :  ENT
   RECIPE:  X-114      ,PANCAKES (MIX)                   ,KIND :  ENT
   RECIPE:  X-42       ,MAPLE SYRUP                      ,KIND :  OTH
   RECIPE:  X-43       ,JAM/JELLY                        ,KIND :  OTH
   RECIPE:  X-44       ,CEREAL READY-TO-EAT              ,KIND :  ENT
   RECIPE:  X-49       ,BUTTER                           ,KIND :  OTH
   RECIPE:  X-50       ,MILK                             ,KIND :  OTH
   RECIPE:  X-62       ,CHILLED CANTALOUPE               ,KIND :  OTH
   RECIPE:  X-78       ,TOAST                            ,KIND :  OTH
   RECIPE:  X-87       ,CHILLED ORANGE JUICE             ,KIND :  OTH

**MENU NUMBER:  B-006     IS SERVED     4 TIMES.**
   RECIPE:  C-12       ,HOT TEA                          ,KIND :  OTH
   RECIPE:  C-5        ,COFFEE AUTOMATIC URN             ,KIND :  OTH
   RECIPE:  D-22       ,FRENCH TOAST                     ,KIND :  ENT
   RECIPE:  F-10       ,GRIDDLE FRIED EGGS               ,KIND :  ENT
   RECIPE:  F-11-2     ,CHEESE OMELETTE                  ,KIND :  ENT
   RECIPE:  F-13       ,SCRAMBLED EGGS                   ,KIND :  ENT
   RECIPE:  L-3        ,GRILLED BACON                    ,KIND :  ENT
   RECIPE:  L-30       ,CRMD GROUND BEEF                 ,KIND :  ENT
   RECIPE:  X-114      ,PANCAKES (MIX)                   ,KIND :  ENT
   RECIPE:  X-23       ,CHILLED APPLEJUICE               ,KIND :  OTH
   RECIPE:  X-42       ,MAPLE SYRUP                      ,KIND :  OTH
   RECIPE:  X-43       ,JAM/JELLY                        ,KIND :  OTH
   RECIPE:  X-44       ,CEREAL READY-TO-EAT              ,KIND :  ENT
   RECIPE:  X-49       ,BUTTER                           ,KIND :  OTH
   RECIPE:  X-50       ,MILK                             ,KIND :  OTH
   RECIPE:  X-78       ,TOAST                            ,KIND :  OTH
   RECIPE:  X-85       ,CHILLED GRAPEFRUIT & ORANGE JU   ,KIND :  OTH

**MENU NUMBER:  B-012     IS SERVED     4 TIMES.**
   RECIPE:  C-12       ,HOT TEA                          ,KIND :  OTH
   RECIPE:  C-5        ,COFFEE AUTOMATIC URN             ,KIND :  OTH
   RECIPE:  D-22       ,FRENCH TOAST                     ,KIND :  ENT
   RECIPE:  E-2-3      ,HOT ROLLED OTTS                  ,KIND :  ENT
   RECIPE:  F-10       ,GRIDDLE FRIED EGGS               ,KIND :  ENT
   RECIPE:  F-11-2     ,CHEESE OMELETTE                  ,KIND :  ENT
   RECIPE:  F-13       ,SCRAMBLED EGGS                   ,KIND :  ENT
   RECIPE:  L-2        ,OVEN FRIED BACON                 ,KIND :  ENT
   RECIPE:  L-88-1     ,BAKED SAUSAGE LINKS              ,KIND :  ENT
   RECIPE:  Q-46-2     ,COTTAGE FRIED POTATOES           ,KIND :  STA
   RECIPE:  X-102      ,CHILLED PRUNES                   ,KIND :  OTH
   RECIPE:  X-114      ,PANCAKES (MIX)                   ,KIND :  ENT
   RECIPE:  X-42       ,MAPLE SYRUP                      ,KIND :  OTH
   RECIPE:  X-43       ,JAM/JELLY                        ,KIND :  OTH
   RECIPE:  X-44       ,CEREAL READY-TO-EAT              ,KIND :  ENT
   RECIPE:  X-49       ,BUTTER                           ,KIND :  OTH
   RECIPE:  X-50       ,MILK                             ,KIND :  OTH
   RECIPE:  X-78       ,TOAST                            ,KIND :  OTH
   RECIPE:  X-87       ,CHILLED ORANGE JUICE             ,KIND :  OTH

**MENU NUMBER:  B-014     IS SERVED     4 TIMES.**
   RECIPE:  C-12       ,HOT TEA                          ,KIND :  OTH
   RECIPE:  C-5        ,COFFEE AUTOMATIC URN             ,KIND :  OTH
   RECIPE:  D-22       ,FRENCH TOAST                     ,KIND :  ENT
   RECIPE:  G-36-1     ,SWEET DOUGH                      ,KIND :  OTH
   RECIPE:  D-42-2     ,CINNAMON SUGAR RAISIN FILLING    ,KIND :  OTH
   RECIPE:  D-53       ,BUTTER WASH                      ,KIND :  OTH
   RECIPE:  E-1-1      ,HOMINY GRITS                     ,KIND :  STA
   RECIPE:  F-10       ,GRIDDLE FRIED EGGS               ,KIND :  ENT
   RECIPE:  F-11-2     ,CHEESE OMELETTE                  ,KIND :  ENT
   RECIPE:  F-13       ,SCRAMBLED EGGS                   ,KIND :  ENT
   RECIPE:  L-2        ,OVEN FRIED BACON                 ,KIND :  ENT
```

Figure 2-33. Menu List with Associated Recipes

```
**RECIPE C-12     (HOT TEA                                  ) IS SERVED IN THE FOLLOWING MENUS:
                  B-003     B-006     B-012     B-014     B-018     B-024     B-035     B-041     B-050     B-051
                  D-001     D-003     D-004     D-010     D-030     D-048     D-052     D-056     D-057     D-058
                  D-062     D-069     D-072     D-077     D-078     D-079     D-090     D-093     D-095     D-100
                  D-105     L-002     L-005     L-015     L-021     L-035     L-045     L-050     L-053     L-059
                  L-063     L-076     L-079     L-093     L-099     L-111
                  TOTAL TIMES SERVED =  112

**RECIPE C-13     (ICED TEA (INSTANT)                       ) IS SERVED IN THE FOLLOWING MENUS:
                  L-065     L-106
                  TOTAL TIMES SERVED =    4

**RECIPE C-5      (COFFEE AUTOMATIC URN                     ) IS SERVED IN THE FOLLOWING MENUS:
                  B-003     B-006     B-012     B-014     B-018     B-024     B-035     B-041     B-050     B-051
                  B-063     D-001     D-003     D-004     D-010     D-030     D-048     D-052     D-056     D-057
                  D-058     D-062     D-069     D-072     D-077     D-078     D-079     D-090     D-093     D-095
                  D-100     D-105     L-002     L-005     L-015     L-020     L-021     L-035     L-045     L-046
                  L-047     L-050     L-053     L-059     L-063     L-065     L-076     L-079     L-093     L-099
                  L-106     L-110     L-111
                  TOTAL TIMES SERVED =  126

**RECIPE C-6      (FRUIT PUNCH                              ) IS SERVED IN THE FOLLOWING MENUS:
                  L-020     L-110
                  TOTAL TIMES SERVED =    4

**RECIPE C-7      (GRAPEADE                                 ) IS SERVED IN THE FOLLOWING MENUS:
                  L-019
                  TOTAL TIMES SERVED =    2

**RECIPE C-8-1    (GRAPE LEMONADE                           ) IS SERVED IN THE FOLLOWING MENUS:
                  L-036
                  TOTAL TIMES SERVED =    2

**RECIPE D-1      (BAKING POWDER BISCUITS                   ) IS SERVED IN THE FOLLOWING MENUS:
                  L-021
                  TOTAL TIMES SERVED =    2

**RECIPE D-13     (QUICK COFFEE CAKE                        ) IS SERVED IN THE FOLLOWING MENUS:
                  B-018
                  TOTAL TIMES SERVED =    4

**RECIPE D-22     (FRENCH TOAST                             ) IS SERVED IN THE FOLLOWING MENUS:
                  B-003     B-006     B-012     B-014     B-018     B-024     B-035     B-041     B-050     B-051
                  B-063
                  TOTAL TIMES SERVED =   42

**RECIPE D-29-1   (BLUEBERRY MUFFINS                        ) IS SERVED IN THE FOLLOWING MENUS:
                  B-003     B-050
                  TOTAL TIMES SERVED =    8

**RECIPE D-31-6   (ITALIAN SAUSAGE PIZZA                    ) IS SERVED IN THE FOLLOWING MENUS:
                  L-079     L-106
                  TOTAL TIMES SERVED =    4

**RECIPE D-33-2   (HOT ROLLS (ROLL MIX)                     ) IS SERVED IN THE FOLLOWING MENUS:
                  D-001     D-003     D-004     D-010     D-030     D-048     D-052     D-056     D-057     D-062
                  D-069     D-072     D-078     D-079     D-090     D-093     D-095     D-100     L-015     L-035
                  L-045     L-059     L-063     L-076     L-093     L-099
                  TOTAL TIMES SERVED =   52

**RECIPE D-36-1   (SWEET DOUGH                             ) IS SERVED IN THE FOLLOWING MENUS:
                  B-014     B-051
                  TOTAL TIMES SERVED =    8

**RECIPE D-42-2   (CINNAMON SUGAR RAISIN FILLING ) IS SERVED IN THE FOLLOWING MENUS:
                  B-014
```

Figure 2-34. Recipe-Menu Cross Reference List

c. Refining the Solution. As mentioned earlier, the process of producing a menu plan is often an iterative one in that there are a number of factors that may be unsatisfactory in the context of the overall plan. The plan may usually be refined by changing selected parameters, and regenerating the solution. The order in which the various parameters are changed can be important, and therefore the following general order is recommended.

- Change priority order until relative goal achievement is satisfactory.
- Adjust goals if desired.
- Adjust bounds.

2-7. SUMMARY. Although this chapter is intended to provide the user with the necessary information to operate the model, the best use of the model may be realized through a complete understanding of the concepts behind the model design. It is therefore recommended that the menu planner use this user's guide in conjunction with the study report. Additional information may also be gained by an understanding of the material to be presented in the next chapter. The most important factor in planning good menus is still experience, just as it has been in the past; therefore, the menu planner should use the model to explore concepts and simply practice generating sample menu plans.

CHAPTER 3

PROGRAMER'S REFERENCE MANUAL

3-1. INTRODUCTION. This chapter is intended to provide the user with program descriptions for each routine in the menu planning model. The information provided in this chapter, when used in conjunction with the documented source code listing, will enable the programer to maintain and modify the programs as needed. The organization of this chapter corresponds to the structure of the model. Each of the three modules is discussed in the order in which they would typically be employed. An overview of the module is followed by file attribute information and a description of each subprogram. The intent is to provide the programer with as much information as is needed to maintain the model. Inconsistencies in format may be a result of the unique nature of some of the programs and the long period of time over which the programs were developed and documented.

3-2. DATA HANDLING MODULE. The data handling module maintains two direct access files: the recipe attribute file and the menu component file. Various operations may be performed on each file including the entry, deletion, and modification of data. Data is located by way of a hashing algorithm with the key being either the recipe number or the menu number, depending on the file. Linear probing is the search procedure that is employed, and the entire file will be searched before indicating that a record cannot be found. A third file, the menu attribute file, is created by the preprocessor, as described in Chapter 2. Listings of each of the files may be produced along with a recipe-menu cross reference listing. These and other files associated with the data module are discussed below. Some of the files, as indicated, are defined for output to the remote terminal and printer.

a. Files

(1) Unit 6

(a) Name. Remote terminal

(b) Purpose. Output to user.

(c) Description. Information is displayed to the user from which decisions may be made concerning potential transactions.

(d) File Statement. FILE 6(KIND="REMOTE",MAXRECSIZE=14)

(2) Unit 10

(a) Name. CAA/RECIPEDATA

(b) Purpose. Recipe attribute data file. Maintains current data concerning recipes and attributes: food cost, labor cost, acceptability, nutrients, and kind.

(c) Description. Direct access file. 4,999 records, each 130 characters (see Table 3-1 for file format).

(d) File Statement.

```
FILE 10(TITLE="CAA/RECIPEDATA",KIND="DISK",MYUSE="IO",
UPDATEFILE="TRUE",INTMODE=4,UNITS="CHARACTERS")
```

Table 3-1. Recipe Attribute File

Field	Columns	Contents	Comments
1	1	Index	Indicate by 0 or 1 whether record is empty or not
2	2-11	Recipe number	As defined in TM 10-412 plus other TSA recipe numbers
3	12-41	Recipe name	Up to 30 characters
4	42-44	Recipe kind	Three-letter abbreviations for entree, vegetable, starch, salad, dessert, other
5	45-50	Food cost	$/100 servings
6	51-55	Labor cost	Manhours/100 servings
7	56-60	Acceptability	Percentage
8	61-67	Calories	Calories/100 servings
9	68-74	Protein	gm/100 servings
10	75-81	Fat	gm/100 servings
11	82-88	Calcium	mg/100 servings
12	89-95	Iron	mg/100 servings
13	96-102	Vitamin A	IU/100 servings
14	103-109	Thiamin	mg/100 servings
15	110-116	Riboflavin	mg/100 servings
16	117-123	Niacin	mg/100 servings
17	124-130	Vitamin C	mg/100 servings

(3) Unit 11

(a) Name. Printer

(b) Purpose. Queue listings from recipe attribute file to printer

(c) Description. Not applicable.

(d) File Statement. FILE 11(KIND="PRINTER")

(4) Unit 12

(a) Name. CAA/MENUDATA

(b) Purpose. Menu component file. Maintains current data concerning menus and recipes that comprise each.

(c) Description. Direct access file. 1,999 records, each 330 characters long. Menu number is followed by the number of recipes and the list of recipe numbers comprising that menu (see Table 3-2 for file format).

(d) File Statement.

```
FILE 12 (TITLE="CAA/MENUDATA.",KIND="DISK",MYUSE="IO",
UPDATEFILE="TRUE",INTMODE=4,UNITS="CHARACTERS")
```

Table 3-2. Menu Component File

Field	Columns	Contents	Comments
1	1	Index	Indicate by 0 or 1 whether record is empty or not
2	2-11	Menu number	Sequentially ordered and preceded by a letter indicating type menu, Ex.: B-001, B-002,...L-001, L-002,...etc.
3	12-13	Number of recipes	The number of recipes in the menu. Maximum = 30
4-33	14-330	Recipe number	Same as recipe file

(5) Unit 13

(a) Name. Printer

(b) Purpose. Queues listings from menu component file to printer.

(c) Description. Not applicable.

(d) File Statement. FILE 13(KIND="PRINTER")

(6) Unit 14

(a) Name. CAA/MENATTDAT

(b) Purpose. Menu attribute data file. Maintains current data concerning menus and attributes: food cost, acceptability, labor, and nutrients.

(c) Description. Sequential file. 130 characters per record. Generated from the menu component file and the recipe attribute files by the preprocessor. Therefore, the number of menus corresponds to the number on the menu component file (see Table 3-3 for file format).

(d) File Statement.

```
FILE 14(TITLE="CAA/MENATTDAT",KIND="DISK",MYUSE="IO",
UPDATEFILE="TRUE",INTMODE=4,UNITS="CHARACTERS")
```

Table 3-3. Menu Attribute File

Field	Columns	Contents	Comments
1	1-10	Menu number	Same as menu component file
2	11-15	Acceptability	Percentage
3	17-22	Food cost	$/100 Servings
4	24-29	Labor cost	Manhours/100 servings
5	31-39	Calories	Calories/100 servings
6	40-48	Protein	gm/100 servings
7	49-57	Fat	gm/100 servings
8	58-66	Calcium	mg/100 servings
9	67-75	Iron	mg/100 servings
10	78-87	Vitamin A	IU/100 servings
11	88-95	Thiamin	mg/100 servings
12	96-103	Riboflavin	mg/100 servings
13	104-111	Niacin	mg/100 servings
14	112-120	Vitamin C	mg/100 servings

(7) Unit 15

(a) Name. Printer.

(b) Purpose. Queues listings of data from the menu attribute file to the printer.

(c) Description. Not applicable.

(d) File Statement. FILE 15(KIND="PRINTER")

(8) Unit 20

(a) Name. Scratch

(b) Purpose. Scratch file for sorts.

(c) Description. Not applicable.

(d) File Statement. FILE 20(STATUS="SCRATCH",MYUSE="IO").

(9) Unit 21

(a) Name. CAA/TSARECDAT

(b) Purpose. Recipe data file from which data may be loaded.

(c) Description. Sequential file. Format corresponds to the original recipe data file provided by TSA. Format may be changed to correspond to appropriate data source (see Table 3-4 for file format).

(d) File Statement.

```
FILE 21 (TITLE="CAA/TSARECDAT.",KIND="DISK",MYUSE="IO",
UPDATEFILE="TRUE",INTMODE=4,UNITS="CHARACTERS")
```

Table 3-4. TSA Recipe Data File

Record	Columns	Contents	Comments
1	1-10	Recipe number	Up to 10 characters
1	16-60	Recipe name	Only first 30 characters will be read
1	73-75	Number of items	Not required for this model
2	1-80	Ingredient data	Not required for this model
3	1-80	Weight & volume requirements	Not required for this model
4	15-18	Acceptability (%)	F 4.0
4	66-70	Labor in manhrs	F 5.2
4	76-80	Food cost in dollars	F 5.2
5	16-21	Calories (kcal)	F 6.0
5	22-27	Protein (gm)	F 6.2
5	28-33	Fat (gm)	F 6.1
5	34-39	Calcium (mg)	F 6.0
5	40-45	Iron (mg)	F 6.1
5	46-51	Vitamin A (IV)	F 6.0
5	52-57	Thiamin (gm)	F 6.2
5	58-63	Riboflavin (gm)	F 6.2
5	64-69	Niacin (gm)	F 6.1
5	70-75	Vitamin C (gm)	F 6.1

(10) Unit 22

(a) Name. CAA/WRKRECDAT

(b) Purpose. Recipe data file from which data may be loaded.

(c) Description. Sequential file. Same format as CAA/RECIPE-DATA. Empty records may be indicated by a zero in first column; occupied records have a 1 in the first column. This format allows the user to copy CAA/RECIPEDATA to CAA/WRKRECDAT for later use.

(d) File Statement.

```
FILE 22(TITLE="CAA/WRKRECDAT",KIND="DISK",MYUSE="IO",
UPDATEFILE="TRUE",INTMODE=4,UNITS="CHARACTERS")
```

(11) Unit 23

(a) Name. CAA/TSAMENDAT

(b) Purpose. Menu data file from which menu component data may be loaded.

(c) Description. Sequential file. Format corresponds to the original menu data file provided by TSA. Format may be changed to correspond to appropriate data sources (see Table 3-5 for file format).

(d) File Statement.

```
FILE 23(TITLE="CAA/TSAMENDAT",KIND="DISK",MYUSE="IO",
UPDATEFILE="TRUE",INTMODE=4,UNITS="CHARACTERS")
```

Table 3-5. TSA Menu Data File

Record	Columns	Contents	Comments
1	11-30	Meal name	Breakfast, Short Order Lunch, or Dinner
1	31-32	Number of recipes	Determines number of records to follow
2	11-20	Recipe number	Up to 10 characters
2	21-65	Recipe name	Only first 30 characters will be read

(12) Unit 24

(a) Name. CAA/WRKMENDAT

(b) Purpose. Menu component data file from which data may be loaded.

(c) Description. Sequential file. Same format as CAA/MENUDATA. Empty records may be indicated with a zero in column one, while occupied records have a 1 in column one.

(d) File Statement.

```
FILE 24(TITLE="CAA/WRKMENDAT",KIND="DISK",MYUSE="IO",
UPDATEFILE="TRUE",INTMODE=4,UNITS="CHARACTERS")
```

b. Subprogram Descriptions. The material presented in this section is intended to provide the programer with a description of each subprogram or included text in the data handling module. The purpose of the routine is provided along with the names of the routines from which the subprogram is called and the names of the routines called from the subprograms. Names of common blocks, included text, and associated FORTRAN I/O units are also provided. Calling arguments are listed as parameters. Other variables are normally documented within the source code listing.

(1) Name. COPY (subroutine)

Purpose:	Copies one item from file X to file Y. Part of the natural merge sort.
Called by:	COPYR, MERGER
Calling sequence:	CALL COPY(X,Y)
Parameters:	X: input file identifier (integer) Y: output file identifier (integer)
Calls:	PFREAD, PFWRIT
Files:	Input: File X Output: File Y
Common blocks:	see NMPROC and PFPROC
Included text:	NMPROC, PFPROC

(2) Name. COPYR (subroutine)

Purpose: As a part of the sort algorithm, this subroutine copies one run from file X to file Y.

Called by: DISTRI, MERGE, MERGER

Calling sequence: CALL COPYR(X,Y)

Parameters: X: input file identifier (integer)
Y: output file identifier (integer)

Calls: COPY

Files: Input: File X
Output: File Y

Common blocks: See NMPROC and PFPROC

Included text: NMPROC, PFPROC

(3) Name. DELMEN (subroutine)

Purpose: This subroutine deletes menus from the menu component file.

Called by: MENU

Calling sequence: CALL DELMEN

Parameters: none

Calls: HASHM

Files: Input: 5, 12
Output: 6, 12

Common blocks: MENCOM

Included text: none

(4) Name. DELREC (subroutine)

Purpose: This subroutine deletes recipes from the recipe attribute file.

Called by: RECIPE

Calling sequence: CALL DELREC

Parameters: none

Calls: HASHR

Files: Input: 5, 10
Output: 6, 10

Common blocks: RECCOM

Included text: none

(5) Name. DESSRT (common block)

Purpose: This common block contains initialized variables pertaining to desserts for use in the preprocessor.

Variables: DESCNT=dessert counter (integer)
DESAC = acceptability of individual dessert (real)
DESACC = acceptability of dessert portion of meal (real)
DESFC = dessert food cost (real)
DESLC = dessert labor cost (real)
DESNUT = array of 10 dessert nutrients (real)
DESDEN = denominator, sum of dessert acceptability (real)

Used in: INITMA, PREP

(6) Name. DISTRI (subroutine)

Purpose: As a part of the sort algorithm, this subroutine distributes runs from file C to files A and B.

Called by: NMSORT

Calling sequence: CALL DISTRI

Parameters: none

Calls: COPYR

Files: This subroutine has no direct input/output

Common blocks: see NMPROC and PFPROC

Included text: NMPROC, PFPROC

(7) Name. ENTREE (common block)

Purpose: This common block contains initialized variables pertaining to entrees for use in the preprocessor.

Variables: ENTCNT = entree counter (integer)
ENTAC = acceptability of individual entree (real)
ENTACC = acceptability of entree portion of meal (real)
ENTFC = entree food cost (real)
ENTLC = entree labor cost (real)
ENTNUT = array of 10 entree nutrients (real)
ENTDEN = denominator, sum of entree acceptability (real)

Used in: INITMA, PREP

(8) Name. EXEC (main program)

Purpose: This is the executive routine for the data handling module. It displays user information.

Called by: Not applicable.

Calling sequence: Not applicable.

Parameters: none

Calls: RECIPE, MENU, PREP, MENATT, XREF

Files: Input: 5
Output: 6

Common blocks: none

Included text: none

(9) Name. GETMEN (subroutine)

Purpose: This subroutine retrieves data from the menu component file.

Called by: MENATT

Calling sequence: CALL GETMEN(MENNUM, NUMREC, RECNUM)

Parameters:
MENNUM: Menu number (Character*10)
NUMREC: Number of recipes (Integer)
RECNUM: Array of up to 30 recipe numbers (Character*10)

Calls: HASHM

Files:
Input: 12
Output: 6

Common blocks: MENCOM

Included text: none

(10) Name. GETREC (subroutine)

Purpose: This subroutine retrieves recipe data from the recipe attribute file.

Called by: LSTXRF, MENATT, PREP

Calling sequence: CALL GETREC(RECNUM, NAME, KIND, RECRFC, RECLC, RECACC, RECNUT)

Parameters:
RECNUM: Recipe number (Character*10)
NAME: Recipe name (Character*30)
KIND: Recipe kind (Character*3)
RECRFC: Recipe labor cost (Real)
RECLC: Recipe food cost (Real)
RECACC: Recipe acceptability (Real)
RECNUT: Array of 10 nutrients (Real)

Calls: HASHR

Files:
Input 10
Output 6

Common blocks: RECCOM

Included text: none

(11) Name. HASHA (function)

Purpose: This function hashes a Character*10 value into an integer value by performing an exclusive-or operation on the bit representation of the first 4 characters and the next 4 characters, then multiplying two 16-bit parts.

Called by: HASHM, HASHR

Calling sequence: HASHA(NUMBER)

Parameters: NUMBER: Number to be hashed (Character*10)

Calls: none

Files: none

Common blocks: none

Included text: none

(12) Name. HASHM (subroutine)

Purpose: Determines the address on the menu component data file for any menu number.

Calledby: DELMEN, GETMEN, INSMEN, LDTSAM, LDWRKM, LOCKMEN, MODMEN

Calling sequence: CALL HASHM(NUMBER, RBA, STOP)

Parameters: NUMBER: Menu number (Character*10)
RBA: Address on the menu component file for any recipe number (Integer)
STOP: Address beyond which no searching will be done (Integer)

Calls: HASHA

Files: none

Common blocks: MENCOM

Included text: none

(13) Name. HASHR (subroutine)

Purpose: Determines the address on the recipe attribute data file for any recipe number.

Called by: DELREC, GETREC, INSREC, LDTSAR, LDWRKR, LOCREC, MODREC

Calling sequence: CALL HASHR(NUMBER, RBA, STOP)

Parameters: NUMBER: Recipe number (Character*10)
RBA: Address on the Recipe attribute file for any recipe number (Integer)
STOP: Address beyond which no searching will be done (Integer)

Calls: HASHA

Files: none

Common blocks: RECCOM

Included text: none

(14) Name. INITM (subroutine)

Purpose: This subroutine will initialize the menu component file for direct access by placing a zero in the first field of each record. This subroutine is used prior to loading data.

Called by: LODMEN

Calling sequence: CALL INITM

Parameters: none

Calls: none

Files: Input: none
Output: 6, 12

Common blocks: MENCOM

Included text: none

(15) Name. INITMA (subroutine)

Purpose: This subroutine initializes the variables that are used in the preprocessor.

Called by: PREP

Calling sequence: CALL INITMA

Parameters: see PREP

Calls: none

Files: none

Common blocks: MENCOM, RECCOM, ENTREE, VEGET, STARCH, SALAD, DESSRT, OTHER, MENUS

Included text: none

(16) Name. INITR (subroutine)

Purpose: This subroutine initializes the recipe attribute file for direct access by placing a zero in the first field of every record. This subroutine is used prior to loading data.

Called by: LODREC

Calling sequence: CALL INITR

Parameters: none

Calls: none

Files: Input: none
Output: 6, 10

Common blocks: RECCOM

Included text: none

(17) Name. INSMEN (subroutine)

Purpose: This subroutine will insert menus and their associated recipes into the menu component data file.

Called by: MENU

Calling sequence: CALL INSMEN

Parameters: none

Calls: HASHM

Files: Input: 5, 12
Output: 6, 12

Common blocks: MENCOM

Included text: none

(18) Name. INSREC (subroutine)

Purpose: This subroutine will insert recipes and their attributes into the recipe attribute data file.

Called by: RECIPE

Calling sequence: CALL INSREC

Parameters: none

Calls: HASHR

Files: none

Common blocks: RECCOM

Included text: none

(19) Name. LDTSAM (subroutine)

Purpose: This subroutine will load a TSA formatted menu data file from unit 23 onto the direct menu component data file on unit 12. This subroutine is meant to allow the user to load menu data from sources such as a management information system.

Called by: LODMEN

Calling sequence: CALL LDTSAM

Parameters: none

Calls: HASHM

Files: Input: 12, 23
Output: 6, 12

Common blocks: MENCOM

Included text: none

(20) Name. LDTSAR (subroutine)

Purpose: This subroutine will load a TSA formatted recipe data file from unit 21 onto the direct recipe data file on unit 10. This enables the user to load recipe data from an externally created data file such as might be created by a management information system.

Called by: LODREC

Calling sequence: CALL LDTSAR

Parameters: none

Calls: HASHR

Files: Input: 10, 21
Output: 6, 10

Common blocks: RECCOM

Included text: none

(21) Name. LDWRKM (subroutine)

Purpose: This subroutine will load a working menu data file from unit 24 onto the direct menu component data file on unit 12. This enables the user to load menu data from a previously prepared data file. As an example, the user may have designed a special set of menus by interfacing with the menu component file. Once that menu component file was satisfactory, it may have been copied out into another file for later use. When the user wants to use that file he or she simply loads it by using this routine.

Called by: LODMEN

Calling sequence: CALL LDWRKM

Parameters: none

Calls: HASHM

Files: Input: 12, 24
Output: 6, 12

Common blocks: MENCOM

Included text: none

(22) Name. LDWRKR (subroutine)

Purpose: This subroutine will load a working recipe data file from unit 22 onto the direct recipe attribute data file on unit 10. This enables the user to load recipe data from a previously prepared data file. As an example, the user may have designed a special set of recipes by interfacing with the recipe attribute file. Once that recipe attribute file was satisfactory, it may have been copied out another file for later use. When the user wants to use that data, he or she simply loads it by using this routine.

Called by: LODREC

Calling sequence: CALL LDWRKR

Parameters: none

Calls: HASHR

Files: Input: 10, 22
Output: 6, 10

Common blocks: RECCOM

Included text: none

(23) Name. LOCMEN (subroutine)

Purpose: This subroutine will locate data concerning individual menus on the menu component data file, and display that data to the user.

Called by: MENU

Calling sequence: CALL LOCMEN

Parameters: none

Calls: HASHM

Files: Input: 5, 12
Output: 6

Common blocks: MENCOM

Included text: none

(24) Name. LOCREC (subroutine)

Purpose: This routine will locate data concerning individual recipes on the recipe attribute data file, and display that data to the user.

Called by: RECIPE

Calling sequence: CALL LOCREC

Parameters: none

Calls: HASHR

Files: Input: 5, 10
Output: 6

Common blocks: RECCOM

Included text: none

(25) Name. LODMEN (subroutine)

Purpose: This subroutine will load an external menu data file. Two type files may be loaded: a TSA formatted menu data file, or a working menu data file with a format corresponding to that of the direct menu component data file. The menu component file is initialized by a call to INITM prior to loading.

Called by: MENU

Calling sequence: CALL LODMEN

Parameters: none

Calls: INITM, LDTSAM, LDWRKM

Files: Input: 5
Output: 6

Common blocks: none

Included text: none

(26) Name. LODREC (subroutine)

Purpose: This subroutine calls other subroutines to load an external recipe data file. Two type files may be loaded: a TSA formatted recipe data file, or a working recipe data file with a format corresponding to that of the direct recipe attribute data file. The recipe attribute file is initialized by a call to INITR prior to loading.

Called by: RECIPE

Calling sequence: CALL LODREC

Parameters: TYPE: Request type (integer)

Calls: INITR, LDTSAR, LDWRKR

Files: Input: 5
Output: 6

Common blocks: none

Included text: none

(27) Name. LSTMEN (subroutine)

Purpose: This subroutine produces listings of selected data from the current menu component data file. These listings are queued to the printer via Unit 13.

Called by: MENU

Calling sequence: CALL LSTMEN

Parameters: none

Calls: SORTM

Files: Input: 5, 12, 20 (scratch file for sorts)
Output: 6, 13, 20 (scratch file for sorts)

Common blocks: MENCOM

Included text: none

(28) Name. LSTREC (subroutine)

Purpose: This subroutine produces listings of selected data from the current recipe attribute data file. These listings are queued to the printer via Unit 11.

Called by: RECIPE

Calling sequence: CALL LSTREC

Parameters: none

Calls: SORTR

Files: Input: 5, 10, 20 (scratch file for sorts)
Output: 6, 11, 20 (scratch file for sorts)

Common blocks: RECCOM

Included text: none

(29) Name. LSTXRF (subroutine)

Purpose: This subroutine will display the cross reference list for all menus and their associated recipes. The total number of times that each recipe appears will also be displayed.

Called by: XREF

Calling sequence: CALL LSTXRF

Parameters: none

Calls: GETREC

Files: Input: 20 (previously sorted file)
Output: 9

Common blocks: RECCOM

Included text: none

(30) Name. MENATT (subroutine)

Purpose: This subroutine produces listings of the menu attribute data file. The listings may be of individual menus, all menus, or all menus of a given meal type.

Called by: EXEC (main program)

Calling sequence: CALL MENATT

Parameters: none

(31) Name. MENCOM (common block)

Calls: GETMEN, GETREC

Files: Input: 5, 14
Output: 6, 15

Common blocks: MENCOM
RECCOM

Included text: none

Purpose: This common block contains the upper limit on the number of menus that can be on the menu component file.

Variables: LASTMN = Maximum number of menus. This integer variable should be set to a prime number. In the model development, it was set to 1999.

Used in: DELMEN, GETMEN, HASHM, INITM, INITMA, INSMEN, LDTSAM, LDWRKM, LOCMEN, LSTMEN, MENATT, MENU, MODMEN

(32) Name. MENU (subroutine)

Purpose: This subroutine is designed to maintain a direct file of menus and associated data. There are six executable options: list, locate, delete, insert, modify, and load. A seventh option terminates the routine.

Called by: EXEC (main program)

Calling sequence: CALL MENU

Parameters: none

Calls: DELMEN, INSMEN, LOCMEN, LODMEN, LSTMEN, MODMEN

Files: Input: 5
Output: 6

Common blocks: MENCOM

Included text: none

(33) Name. MENUS (common block)

Purpose: This common block contains initialized variables pertaining to menu attributes for use in the preprocessor.

Variables: MENRFC = menu food cost (real)
MENLC = menu labor cost (real)
MENNUT = array of 10 nutrients for the menu (real)

Used in: INITMA, PREP

(34) Name. MERGE (subroutine)

Purpose: Part of the sort algorithm, this subroutine merges runs from files A and B into file C.

Called by: NMSORT

Calling sequence: CALL MERGE

Parameters: none

Calls: COPYR, MERGER

Files: No direct input/output

Common blocks: see NMPROC and PFROC

Included test: NMPROC, PFPROC

(35) Name. MERGER (subroutine)

Purpose: Part of the sort algorithm, this subroutine will merge a pair of runs, one from A and one from B, into a single run on C.

Called by: MERGE

Calling sequence: CALL MERGER

Parameters: none

Calls: COPY, COPYR

Common blocks: see NMPROC and PFPROC

Included text: NMPROC, PFPROC

(36) Name. MODMEN (subroutine)

Purpose: This subroutine will make selected changes to current menus.

Called by: MENU

Calling sequence: CALL MODMEN

Parameters: none

Calls: HASHM

Files: Input: 5, 12
Output: 6, 12

Common blocks: MENCOM

Included text: none

(37) Name. MODREC (subroutine)

Purpose: This subroutine will make selected changes to current recipes.

Called by: RECIPE

Calling sequence: CALL MODREC

Parameters: none

Calls: HASHR

Files: Input: 5, 10
Output: 6, 10

Common blocks: RECCOM

Included text: none

(38) Name. NMPROC (included text)

Purpose: This procedure is included in subroutines that are part of the sort algorithm. The purpose of the procedure is to establish parameters and common blocks. Parameters are explained in the documented source code listing.

Called by: Not applicable.

Calling sequence: Not applicable.

FORTRAN parameters (i.e., symbolic names for constants):
A,B,C = FILE NUMBERS FOR 'PASCAL'- STYLE 10 (integers)

Calls: none

Files: none

Common blocks: /NMDAT/ L, EOR, RLEN, KSTART, KEND
L = THE NUMBER OF RUNS MERGED (integer)
EOR = END-OF-RUN INDICATOR (integer)
RLEN = LENGTH OF RECORDS (integer)
KSTART = STARTING CHARACTER POSITION OF THE SORT KEY (integer)
KEND = ENDING CHARACTER POSITION OF THE SORT KEY

Included text: none

(39) Name. NMSORT (subroutine)

Purpose: This subroutine is the main subroutine for the sort algorithm. It validates input parameters and begins natural merge sort.

Called by: LSTREC

Calling sequence: CALL NMSORT(IUNITC, IUNITA, IUNITB, IRECL, IKSTRT, IKEND)

Parameters: IUNITC = THE FORTRAN unit containing the input data. On completion of the sort, the sorted data has replaced the input data (integer).
IUNITA, IUNITB = Two FORTRAN unit numbers for work files. (Note: all units must be capable of being rewound and rewritten) (integer)
IRECL = The length (characters) of the input records (integer)
IKSTRT = The starting position (characters) of the key field (integer).
IKEND = The ending position (characters) of the key fields (integer)

Calls: DISTRI, MERGE, PFRSET, PFRWRT

Files: This subroutine has no direct input/output but calls subroutines that read from and write to files IUNITA, IUNITB, and IUNITC.

Common blocks: see NMPROC and PFPROC

Included text: NMPROC, PFPROC

(40) Name. OTHER (common block)

Purpose: This common block contains initialized variables pertaining to "other" recipe and menus for use in the preprocessor.

Variables: OTHFC = "other" food cost (real)
OTHLC = "other" labor cost (real)
OTHNUT = array of 10 "other" nutrients (real)

Used in: INITMA, PREP

(41) Name. PFERR (subroutine)

Purpose: As a part of the sort algorithm, this subroutine issues an error message and stops. The message is selected from an interval table of "canned" messages.

Called by: PFREAD, PFRSET, PFRWRT, PFWRIT

Calling sequence: CALL PFERR(MSGNO)

Parameters: MSGNO: A number corresponding to the error to be displayed (integer)

Calls: none

Files: Output: 6

Common blocks: none

Included text:

(42) Name. PFPROC (included text)

Purpose: This procedure is included in subroutines that are part of the sort algorithm. The purpose of the procedure is to establish parameters and common blocks.

Called by: Not applicable.

Calling sequence: Not applicable.

FORTRAN parameters (i.e., symbolic names for constants):
MAXLEN = The maximum length record that can be processed (integer)
MAXF = The maximum number of files that can be handled (integer)

Calls: none

Files: none

Common blocks: /PFCDAT/ BUFFER(MAXF)
/PFDAT/ UNIT(MAXF), RECLEN(MAXF), EOF(MAXF)
BUFFER(I) = the input buffer for file I (character * maxlen)
UNIT(I) = The FORTRAN unit number for file I (integer)
RECLEN(I) = The length of the record to be read from file I (integer)
EOF(I) = The (PASCAL) end-of-file indicator for file I (logical)

Included text: none

(43) Name. PFREAD (subroutine)

Purpose: As a part of the sort algorithm, this subroutine simulates a PASCAL style "READ (F,X)"

Called by: COPY

Calling sequence: CALL PFREAD(F,X)

Parameters: F: input file identifier (integer)
X: input record (character)

Calls: PFERR

Files: A, B, or C

Common blocks: See PFPROC

Included text: PFPROC

(44) Name. PFRSET (subroutine)

Purpose: As a part of the sort algorithm, this subroutine resets file F by rewinding and reading first record. This simulates a Pascal RESET(F) instruction.

Called by: NMSORT

Calling sequence: CALL PFRSET(F)

Parameters: F: file identifier (integer)

Calls: PFERR

Files: A, B, or C

Common blocks: See PFPROC

Included text: PFPROC

(45) Name. PFRWRT (subroutine)

Purpose: As a part of the sort algorithm, this subroutine has the purpose of rewriting file F. This simulates a Pascal REWRITE(F) instruction.

Called by: NMSORT

Calling sequence: CALL PFRWRT(F)

Parameters: F: File identifier (integer)

Calls: PFERR

Files: A, B, or C

Common blocks: See PFPROC

Included text: PFPROC

(46) Name. PFWRIT (subroutine)

Purpose: As part of the sort algorithm, this subroutine simulates a PASCAL TYPE "WRITE(F,X)"

Called by: COPY

Calling sequence: CALL PFWRIT(F,X)

Parameters: F: output file identifier (integer)
X: output record (character)

Calls: PFERR

Files: A, B, or C

Common blocks: See PFPROC

Included text: PFPROC NONE

(47) Name. PREP (subroutine)

Purpose: This subroutine is intended to generate the menu attribute data from the menu component data and the recipe attribute data. The subroutine accesses the menu file sequentially, and retrieves recipe attribute data for each recipe in that particular menu. When all recipe data has been gathered, the menu attributes of food cost, labor cost, nutrition and acceptability are computed and then written on the menu attribute data file. When all menus have been processed, the file is sorted by menu number.

Called by: EXEC (main program)

Calling sequence: CALL PREP

Parameters: none

Calls: [illegible]ORT, GETREC, INITMA

Files: Input: 12
Output: 6, 14

Common blocks: MENCOM, RECCOM, ENTREE, VEGET, STARCH, SALAD, DESSRT, OTHER, MENUS

Included text: none

(48) Name. RECCOM (common block)

Purpose: This common block contains the upper limit on the number of recipes that can be on the recipe attribute file.

Variables: LASTRC = Maximum number of recipes. This integer variable should be set to a prime number. In the model development, it was set to 4999.

Used in: DELREC, GETREC, HASHR, INITMA, INITR, INSREC, LDTSAR, LDWRKR, LOCREC, LSTREC, LSTXRF, MENATT, MODREC, RECIPE, XREF

(49) Name. RECIPE (subroutine)

Purpose: This subroutine is designed to maintain a direct file of recipes and their associated attributes. There are seven executable options: list, locate, insert, delete, modify, load, and terminate.

Called by: EXEC (main program)

Calling sequence: CALL RECIPE

Parameters: none

Calls: Calls: DELREC, INSREC, LOCREC, LODREC, LSTREC, MODREC

Files: Input: 5
Output: 6

Common blocks: RECCOM

Included text: none

(50) Name. SALAD (common block)

Purpose: This common block contains initialized variables pertaining to salads for use in the preprocessor.

Variables: SALCNT = salad counter (integer)
SALAC = acceptability of individual salad (real)
SALACC = acceptability of salad portion of meal (real)
SALFC = salad food cost (real)
SALLC - salad labor cost (real)
SALNUT = array of 10 salad nutrients (real)
SALDEN = denominator, sum of salad acceptability (real)

Used in: INITMA, PREP

(51) Name. STARCH (common block)

Purpose: This common block contains initialized variables pertaining to starches for use in the preprocessor.

Variables:
STACNT = starch counter (integer)
STAAC = acceptability of individual starch (real)
STAACC = acceptability of starch portion of meal (real)
STAFC = starch food cost (real)
STALC = starch labor cost (real)
STANUT = array of 10 starch nutrients (real)
STADEN = denominator, sum of starch acceptability (real)

Used in: INITMA, PREP

(52) Name. VEGET (common block)

Purpose: This common block contains initialized variables, pertaining to vegetables for use in the preprocessor.

Variables:
VEGCNT = vegetable counter (integer)
VEGAC = acceptability of individual vegetable (real)
VEGACC = acceptability of vegetable portion of meal (real)
VEGFC = vegetable food cost (integer)
VEGLC = vegetable labor cost (real)
VEGNUT = array of 10 vegetable nutrients (real)
VEGDEN = denominator, sum of vegetable acceptability (real)

Used in: INITMA, PREP

(53) Name. XREF (subroutine)

Purpose:	This subroutine will produce a cross reference listing of recipes and the menus in which they appear for all menus that are listed in the menu component file.
Called by:	EXEC (main program)
Calling sequence:	CALL XREF
Parameters:	none
Calls:	Calls: LSTXRF, SORTX
Files:	Input: 12 Output: 6, 9, 20 (scratch file for sorts)
Common blocks:	RECCOM
Included text:	none

c. Sort Routines. Because data on the recipe attribute and menu component data files are not in sorted order, the routines that display listings from those files call a sort routine prior to sending data to the printer. The preprocessor calls a sort routine before writing data to the menu attribute file. In addition, the logic employed in generating the cross reference list is based on the use of sorted files. The absence of a FORTRAN callable system sort at the time the menu planning model was placed on the Burroughs meant that most output was not displayed in sorted order. An alternative approach to generating the cross reference listing was discussed in Chapter 2. Once a FORTRAN callable system sort is developed, it may be easily incorporated into the model.

As the model is designed, a call to SORTR is intended to sort recipe data records of 130 characters by recipe number (character positions 2 through 11). A call to SORTM is intended to sort menu data records of 323 characters by menu number (character positions 2 through 11). A call to SORTX is intended to sort first by recipe number (character positions 15 through 24), and then by menu number (character positions 1 through 10). The scratch file for sorts is unit 20. Until a FORTRAN callable system sort is incorporated into the model, a natural merge sort may be used. This sort routine was partially incorporated into the model, and the subprograms are described above. The routine may be called by calling NMSORT. In this case, files 9, 20, and 30 are scratch files with unit 20 being the file that is to be sorted.

3-3. PARAMETERIZATION MODULE. The parameterization module enables the user to establish menu planning parameters and to access information regarding those parameters. A matrix generator creates the goal programing problem matrix and the set of initial upper bounds. User interfaces with the bounds, goals, and priority order files are provided. Files associated with the parameterization module are discussed below.

a. Files

(1) Unit 6

(a) Name. Remote terminal

(b) Purpose. Output to user

(c) Description. Output to the user is displayed at the terminal.

(d) File Statement. FILE 6(KIND="REMOTE",MAXRECSIZE=14)

(2) Unit 14

(a) Name. CAA/MENATTDAT

(b) Purpose. Maintains menu attribute data.

(c) Description. See paragraph 3-2a(6).

(d) File Statement.

```
FILE 14(TITLE="CAA/MENATTDAT",KIND="DISK",MYUSE="IO",
UPDATEFILE="TRUE",INTMODE=4,UNITS="CHARACTERS")
```

(3) Unit 16

(a) Name. CAA/GPDATA

(b) Purpose. Maintain goal programing problem matrix.

(c) Description. Sequential file. Data is organized into "ROWS" and "COLUMNS" sections in keeping with the format of FMPS. ROWS section contains all row names including objective function names. COLUMNS section includes column or variable name and the associated row name with the value of the coefficient.

(d) File Statement.

```
FILE  16(TITLE="CAA/GPDATA.",INTMODE=4,KIND="DISK",
UNITS="CHARACTERS",MYUSE="IO",UPDATEFILE="TRUE")
```

(4) Unit 17

(a) Name. CAA/BOUNDS

(b) Purpose. Maintain the bounds portion of the problem.

(c) Description. Sequential file. Data includes all bounded variables, the type bound (LO for lower, FX for fixed, and UP for upper), and the value of the bound.

(d) File Statement.

```
FILE 17(TITLE="CAA/BOUNDS.",INTMODE=4,KIND="DISK",
UNITS="CHARACTERS",MYUSE="IO",UPDATEFILE="TRUE")
```

(5) Unit 18

(a) Name. CAA/RHS

(b) Purpose. Maintain the right-hand side (RHS) portion of the problem.

(c) Description. Sequential file. Data includes row names and the associated RHS values.

(d) File Statement.

```
FILE 18(TITLE="CAA/RHS.",INTMODE=4,KIND="DISK",
UNITS="CHARACTERS",MYUSE="IO",UPDATEFILE="TRUE")
```

(6) Unit 19

(a) Name. CAA/PRIORS

(b) Purpose. Maintains the priority ordering for the attributes.

(c) Description. Sequential file. The name of each objective function is listed in the order in which they are prioritized. Each objective function is followed by the number of constraints associated with that objective and the row names of those constraints.

(d) File Statement.

```
FILE  19(TITLE="CAA/PRIORS.",INTMODE=4,KIND="DISK",
UNITS="CHARACTERS",MYUSE="IO",UPDATEFILE="TRUE")
```

b. Subprogram Descriptions. The material presented in this section is intended to provide the programer with a description of each subprogram or included text in the parameterization module. The purpose of the routine is provided along with the names of the routines from which the subprogram is called and the names of the routines called from the subprograms. Names of common blocks, included text, and associated FORTRAN I/O units are also provided. Calling arguments are listed as parameters. Other variables are normally documented within the source code listing.

(1) Name. BOUNDS (subroutine)

Purpose: This subroutine will allow the user to revise bounds on variables before running the goal programing algorithm. A variable may have one of three type bounds: upper, lower, or fixed. An upper bound is originally placed on each variable by the matrix generator. An upper bound on a variable means that variable may not exceed the value of the bound while a lower bound means that the variable may not take on a value less than the bound. A fixed bound means that the variable must equal the value of the bound. The goal programing algorithm recognizes these bounds as UP, LO, or FX.

Called by: EXEC (main program)

Calling sequence: CALL BOUNDS

Parameters: none

(2) Name. EXEC (main program)

Purpose: This program is the executive routine for the parameterization module. Instructions to the user are displayed upon execution of this routine.

Called by: Not applicable.

Calling sequence: Not applicable.

Parameters: none

Calls: MATGEN, BOUNDS, GOAL, PRIORS

Files: Input: 5
Output: 6

Common blocks: none

Included text: none

(3) Name. FBNDO (subroutine)

Purpose: Writes an FMPS BOUNDS header card

Called by: GEND

Calling sequence: CALL FBNDO

Parameters: none

Calls: none

Files: 16 (output)

Common blocks: none

Included text: none

(4) Name. FBOUND (subroutine)

Purpose: This routine writes an LP upper bound data card in UNIVAC FMPS format to the bounds file. Note that this is not the same file as the LP data file which most of the other routines write to.

Called by: GBOUND

Calling sequence: CALL FBOUND (COLNM, VALUE)

Parameters: COLNM - the column name (Character*8)
VALUE - the upper bound value (real)

Calls: none

Files: 17 (output)

Common blocks: none

Included text: none

(5) Name. FCOL (subroutine)

Purpose: This routine writes an LP column data card, in UNIVAC FMPS format, to the LP data file.

Called by: GACC, GADEV, GFCOST, GFTCAL, GLCOST, GNTOT, GNUTR, GSTRUC

Calling sequence: CALL FCOL (COLNM, ROWNM, VALUE)

Parameters: COLNM - the column name (Character*8)
ROWNM - the row name (Character*8)
VALUE - the coefficient value (real)

Calls: none

Files: 16 (output)

Common blocks: none

Included text: none

(6) Name. FCOLO (subroutine)

Purpose: This routine writes an LP data column section header card, in UNIVAC FMPS format, to the LP data file.

Called by: GSTART

Calling sequence: CALL FCOLO

Parameters: none

Calls: none

Files: 16 (output)

Common blocks: none

Included text: none

(7) Name. FEND (subroutine)

Purpose: This routine writes an LP data 'ENDATA' card, in UNIVAC FMPS format, to the LP data file.

Called by: GEND

Calling sequence: CALL FEND

Parameters: none

Calls: none

Files: 17 (output)

Common blocks: none

Included text: none

(8) Name. FNAME (subroutine)

Purpose: This routine writes an LP data 'NAME' card, in UNIVAC FMPS format, to the LP data file.

Called by: GSTART

Calling sequence: CALL FNAME(NAME)

Parameters: NAME - the name of the data file (Character*8)

Calls: none

Files: 16 (output)

Common blocks: none

Included text: none

(9) Name. FRHSO (subroutine)

Purpose: This routine writes an LP data 'RHS' card, in UNIVAC FMPS format, to the LP data file. In addition, it writes 'ADD RHS' as a flag to indicate that RHS data is to be retrieved.

Called by: GEND

Calling sequence: CALL FRHSO

Parameters: none

Calls: none

Files: 16 (output)

Common blocks: none

Included text: none

(10) Name. FROW (subroutine)

Purpose: This routine writes an LP row data card, in UNIVAC FMPS format, to the LP data file.

Called by: GROWS

Calling sequence: CALL FROW (TYPE, ROWNM)

Parameters: TYPE - the row type (Character*1)
ROWNM - the row name (Character*8)

Calls: none

Files: 16 (output)

Common blocks: none

Included text: none

(11) Name. FROWO (subroutine)

Purpose: This routine writes an LP data row section header card, in UNIVAC FMPS format, to the LP data file.

Called by: GSTART

Calling sequence: CALL FROWO

Parameters: none

Calls: none

Files: 16 (output)

Common blocks: none

Included text: none

(12) Name. GACC (subroutine)

Purpose: This routine generates the column entry in the acceptability goal row for a specific meal and menu.

Called by: MATGEN

Calling sequence: CALL GACC (MEAL, MENNUM, MENACC)

Parameters: MEAL - the meal number (integer)
MENNUM - the menu number (Character*8)
MENACC - the acceptablity of the menu (real)

Calls: FCOL, RAC

Files: none

Common blocks: none

Included text: none

(13) Name. GADEV (subroutine)

Purpose: This routine generates the column entries for the goal programing deviational variables for a specific row in the LP data file. The routine also generates the column entries for these variables in a specific objective function row, with weights as specified by the caller of the routine.

Called by: GDEVS

Calling sequence: CALL GADEV (ROWNM, OBJNM, NEGWT, POSWT)

Parameters: ROWNM - the name of the row
OBJNM - the name of the objective row
NEGWT - the weight to be assigned to the negative deviation
POSWT - the weight to be assigned to the positive deviation

Calls: FCOL

Files: none

Common blocks: none

Included text: none

(14) Name. GBOUND (subroutine)

Purpose: This routine generates an upper bound on a specific menu in a particular meal.

Called by: MATGEN

Calling sequence: CALL GBOUND (MEAL, MENNUM, LIMITS)

Parameters: MEAL - the meal number (Integer)
MENNUM - the menu number (Character*8)
LIMITS - an array containing the upper limit on each meal (Real) dimensioned (4)

Calls: FBOUND

Files: none

Common blocks: none

Included text: none

(15) Name. GDEVS (subroutine)

Purpose: This routine generates the column entries for goal programing deviational variables in the LP data file.

Called by: GSTART

Calling sequence: CALL GDEVS

Parameters: none

Calls: GADEV, RAC, RFAT, RFC, RLC, RNUT, RST

Files: none

Common blocks: none

Included text: none

(16) Name. GEND (subroutine)

Purpose: This routine generates the end of the LP data file.

Called by: MATGEN

Calling sequence: CALL GEND

Parameters: none

Calls: FBNDO, FEND, FRHSO

Files: none

Common blocks: none

Included text: none

(17) Name. GFCOST (subroutine)

Purpose: This routine generates the column entry for the food cost goal row for a specific meal and menu.

Called by: MATGEN

Calling sequence: CALL GFCOST (MEAL,MENNUM,MENRFC)

Parameters: MEAL - the meal number (Integer)
MENNUM - the menu number (Character*8)
MENRFC - the food cost of the menu (Real)

Calls: FCOL, RFC

Files: none

Common blocks: none

Included text: none

(18) Name. GFTCAL (subroutine)

Purpose: This routine generates the column entry for the fat/ calorie ratio goal for a specific meal and menu.

Called by: MATGEN

Calling sequence: CALL GFTCAL (MEAL, MENNUM, CAL, FAT)

Parameters:
MEAL - the meal number (integer)
MENNUM - the menu number (Character*8)
CAL - the number of calories in the menu (real)
FAT - the amount of fat in the menu (real)

Calls: FCOL, RFAT

Files: none

Common blocks: none

Included text: none

(19) Name. GLCOST (subroutine)

Purpose: This routine generates the column entry for the labor cost goal row for a specific meal and menu.

Called by: MATGEN

Calling sequence: CALL GLCOST (MEAL,MENNUM,MENLC)

Parameters:
MEAL - the meal number (integer)
MENNUM - the menu number (Character*8)
M_NLC - the menu labor cost (real)

Calls: FCOL, RLC

Files: none

Common blocks: none

Included text: none

(20) Name. GNTOT (subroutine)

Purpose: This routine generates column entries for goals of nutritional totals for each meal.

Called by: none

Calling sequence: CALL GNTOT

Parameters: none

Calls: FCOL, RNTOTM, RNUT, RNUTM

Files: none

Common blocks: none

Included text: none

(21) Name. GNUTR (subroutine)

Purpose: This routine generates the column entry for the nutrient goal row for a specific combination of meal, menu, and nutrient.

Called by: MATGEN

Calling sequence: CALL GNUTR (MEAL,MENNUM,MENNUT,I)

Parameters: MEAL - the meal number (integer)
MENNUM - the menu number (Character*8)
MENNUT - the amount of the specified nutrient in this menu (Real)
I - the number (1-10) of the nutrient (integer)

Calls: FCOL, RNUT

Files: none

Common blocks: none

Included text: none

(22) Name. GOAL (subroutine)

Purpose: This subroutine allows the user to access the goals (also called the right-hand side values). These values are located on unit 18 in a form that is suitable for access by the goal programing algorithm. The user does not normally see the values on unit 18, but instead sees the values in terms that have more meaning to him or her. Food cost goals are determined from the BDFA, labor cost goals are determined from user input for the number of manhours per meal, acceptability goals are determined from user input for acceptability per meal, nutritional goals are determined from user input for the various nutrients per individual per day (normally the nutrition goals input by the user are the recommended daily food allowances), and the length of the menu planning cycle in days is used to determine the actual goals used by the goal programing algorithm.

Called by: EXEC

Calling sequence: CALL GOAL

Parameters: none

Calls: none

Files: Input: 5, 18
Output: 6, 18

Common blocks: none

Included text: none

(23) Name. GROWS (subroutine)

Purpose: This routine generates the 'ROWS' section of the LP data file.

Called by: GSTART

Calling sequence: CALL GROWS

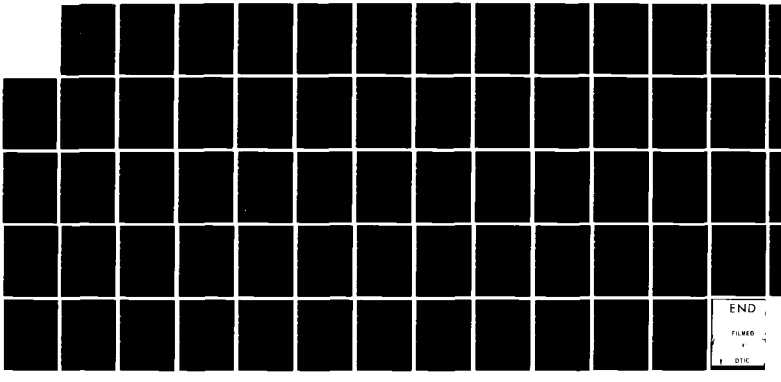
Parameters: none

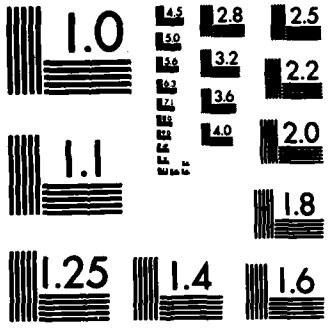
Calls: FROW, RAC, RFAT, RFC, RLC, RNUT

Files: none

AD-A123 225 ECONOMETRIC MODEL FOR OPTIMIZING TROOP DINING FACILITY OPERATIONS USER'S A.. (U) ARMY CONCEPTS ANALYSIS AGENCY BETHESDA MD A C MANGUSO DEC 82 CAA-D-82-4 2/2

UNCLASSIFIED F/G 8/6 NL





Common blocks: none

Included text: none

(24) Name. GSTART (subroutine)

Purpose: This routine initiates the generation of the LP data file.

Called by: MATGEN

Calling sequence: CALL GSTART

Parameters: none

Calls: FCOLO, FNAME, FROWO, GDEVS, GROWS

Files: none

Common blocks: none

Included text: none

(25) Name. GSTRUC (subroutine)

Purpose: This routine generates the column entries for structural goals.

Called by: MATGEN

Calling sequence: CALL GSTRUC (MEAL, MENNUM)

Parameters: MEAL - the meal number (integer)
MENNUM - the menu number (Character*8)

Calls: FCOL, RST

Files: none

Common blocks: none

Included text: none

(26) Name. MATGEN (subroutine)

Purpose: This routine serves to organize the flow of control through the entire LP matrix generation process, obtaining the attributes of each menu in turn, and generating all the required LP data for that menu.

Called by: EXEC

Calling sequence: CALL MATGEN

Parameters: none

Calls: GACC, GBOUND, GEND, GFOCST, GFTCAL, GLCOST, GNUTR, GSTART, GSTRUC, MGET, UGET, UPUT, MGET, UGET, UPUT

Files: 6 (output)

Common blocks: UDATA

Included text: none

(27) Name. MGET (subroutine)

Purpose: This routine gets menu attribute data from the menu attribute data file.

Called by: MATGEN

Calling sequence: CALL MGET(MEAL,MENNUM,MENACC,MENRFC,MENLC, MENNUT,EOMEN)

Output parameters: MEAL - meal number (integer)
MENNUM - menu number (Character*8)
MENACC - menu acceptability (real)
MENRFC - menu food cost (real)
MENLC - menu labor cost (real)
MENNUT - array of menu nutrient contents (real), dimensioned (10)
EOMEN - end of file indicator (logical)

Calls: none

Files: 14 (input)

Common blocks: none

Included text: none

(28) Name. PRIORS (subroutine)

Purpose: This subroutine will allow the user to change the order in which the four attributes are prioritized. The new priority ordering is displayed to the user and the appropriate objective and constraint row data for by the XMP goal programing algorithm are written to unit 19.

Called by: EXEC (main program)

Calling sequence: CALL PRIORS

Parameters: none

Calls: none

Files: Input: 5
Output: 6, 19

Common blocks: none

Included text: none

(29) Name. RAC (function)

Purpose: This function provides the name of the constraint row for acceptability for a specified meal.

Called by: GACC, GDEVS, GROWS

Calling sequence: RAC(M)

Parameters: M - meal number (integer)

Calls: none

Files: none

Common blocks: none

Included text: none

(30) Name. RFAT (function)

Purpose: This function provides the name of the constraint row for the fat/calorie ratio goal for a specified meal.

Called by: GDEVS, GFTCAL, GROWS

Calling sequence: RFAT(M)

Parameters: M - meal number (integer)

Calls: none

Files: none

Common blocks: none

Included text: none

(31) Name. RFC (function)

Purpose: This function provides the name of the constraint row for food cost for a specified meal.

Called by: GDEVS, GFCOST, GROWS

Calling sequence: RFC(M)

Parameters: M - meal number (integer)

Calls: none

Files: none

Common blocks: none

Included text: none

(32) Name. RLC (function)

Purpose: This function provides the name of the constraint row for labor cost for a specified meal.

Called by: GDEVS, GLCOST, GROWS

Calling sequence: RLC(M)

Parameters: M - meal number (integer)

Calls: none

Files: none

Common blocks: none

Included text: none

(33) Name. RNTOTM (function)

Purpose: This routine provides the row name for the total nutritional goal for a specified meal and nutrient.

Called by: GNTOT

Calling sequence: RNTOTM(I,J)

Parameters: I - meal number (integer)
J - nutrient number (integer)

Calls: none

Files: none

Common blocks: none

Included text: none

(34) Name. RNUT (function)

Purpose: This function provides the name of the constraint row for the total nutrition supplied by a specifted nutrient.

Called by: GDEVS, GNTOT, GNUTR, GROWS

Calling sequence: RNUT(I)

Parameters: I - nutrient number (integer)

Calls: none

Files: none

Common blocks: none

Included text: none

(35) Name. RNUTM (function)

Purpose: This function provides the name of the constraint row for the nutritional goal for a specified meal and nutrient.

Called by: GNTOT

Calling sequence: RNUTM(I,J)

Parameters: I - meal number (integer)
J - nutrient number (integer)

Calls: none

Files: none

Common blocks: none

Included text: none

(36) Name. RST (function)

Purpose: This function provides the name of the constraint row for the structural goal for a specified meal.

Called by: GDEVS, GROWS, GSTRUC

Calling sequence: RST(M)

Parameters: M - meal number (integer)

Calls: none

Files: none

Common blocks: none

Included text: none

(37) Name. UDATA (common block)

Purpose: This common block contains the upper limits on menus for each meal.

Variables: LIMITS - (real) the array of upper limits, dimensioned (4)

Used in: MATGEN, UGET, UPUT

(38) Name. UGET (subroutine)

Purpose: This routine queries the user for upper bounds on the number of times any menu may be used for each meal.

Called by: MATGEN

Calling sequence: CALL UGET

Parameters: none

Calls: none

Files: 5 (input)

Common blocks: UDATA

Included text: none

(39) Name. UPUT (subroutine)

Purpose: This routine echoes the upper bounds supplied by the user (via subroutine UGET).

Called by: MATGEN

Calling sequence: CALL UPUT

Parameters: none

Calls: none

Files: 6 (output)

Common blocks: UDATA

Included text: none

3-4. SOLUTION MODULE. The solution module consists of input routines, a solution algorithm (XMP), and a postprocessor. The XMP subprograms will not be discussed in this chapter. Instead, an introduction to XMP is included in Appendix B. The input routines provide an interface between those files produced by the parameterization module and XMP. The postprocessor displays information regarding solutions in a series of five reports. Files associated with the solution module are discussed below. Some of the files, as indicated, are defined for output to the remote terminal and printer.

a. Files

(1) Unit 6

(a) Name. Printer.

(b) Purpose. Solution module produces reports that are too long to be displayed at the terminal and therefore, they are queued to the printer.

(c) Description. Messages, diagnostics, and solution reports.

(d) File Statement. FILE 6 (KIND="PRINTER",MAXRECSIZE=22)

(2) Unit 8

(a) Name. Not applicable.

(b) Purpose. Output file for XMP.

(c) Description. Contains XMP output, including diagnostic and error messages. Solutions for each priority level are listed.

(d) File Statement.

```
FILE  8(STATUS="NEW",MYUSE="IO",UPDATEFILE="TRUE")
```

(3) Unit 9

(a) Name. Not applicable.

(b) Purpose. Scratch file for natural-merge sorts.

(c) Description. If natural merge sort routine is used in place of a FORTRAN callable system sort, this is one of two scratch files used.

(d) File Statement.

```
FILE 9 (STATUS="NEW",MYUSE="IO,UPDATEFILE="TRUE")
```

(4) Unit 10. See para 3-2a(2).

(5) Unit 12. See para 3-2a(4).

(6) Unit 14. See para 3-2a(6).

(7) Unit 16. See para 3-3a(3).

(8) Unit 17. See para 3-3a(4).

(9) Unit 18. See para 3-3a(5).

(10) Unit 19. See para 3-3a(6).

(11) Unit 25

(a) Name. Not applicable.

(b) Purpose. Scratch file for postprocessor.

(c) Description. Contains name, status, and value of deviation variable.

(d) File Statement.

```
FILE 25(STATUS="NEW",MYUSE="IO", UPDATEFILE="TRUE")
```

(12) Unit 26

(a) Name. Not applicable.

(b) Purpose. Scratch file for postprocessor.

(c) Description. Contains menu number, status and value for those menus in the solution.

(d) File Statement.

```
FILE 26(STATUS="NEW",MYUSE="IO", UPDATEFILE="TRUE")
```

(13) Unit 27

(a) Name. Not applicable.

(b) Purpose. Scratch file for postprocessor.

(c) Description. Contains, for each menu in the solution, menu number, number of recipes in the menu, and recipe number of each recipe in the menu.

(d) File Statement.

```
FILE 27(STATUS="NEW",MYUSE="IO",UPDATEFILE="TRUE")
```

(14) Unit 29

(a) Name. Not applicable.

(b) Purpose. Scratch file for postprocessor.

(c) Description. Contains, for each menu in the solution, acceptability, food cost, labor cost, and nutritional value.

(d) File Statement.

```
FILE 28(STATUS="NEW",MYUSE="IO,UPDATEFILE="TRUE")
```

(15) Unit 29

(a) Name. Not applicable.

(b) Purpose. Scratch file for cross reference.

(c) Description. Contains, for each menu in the solution, menu number, value, and recipe numbers of recipes in that menu.

(d) File Statement.

```
FILE 29(STATUS="NEW",MYUSE="IO",UPDATEFILE="TRUE")
```

(16) Unit 30

(a) Name. Not applicable.

(b) Purpose. Scratch file for natural merge sort.

(c) Description. If natural merge sort routine is use in place of a FORTRAN callable system sort, this is one of two scratch files used.

(d) File Statement.

```
FILE 30(STATUS="NEW",MYUSE="IO",UPDATEFILE="TRUE")
```

b. Subprogram Descriptions. The material presented in this section is intended to provide the programer with a description of each subprogram or included text in the solution module. The purpose of the routine is provided along with the names of the routines from which the subprogram is called and the names of the routines called from the subprograms. Names of common blocks, included text, and associated FORTRAN I/O units are also provided. Calling arguments are listed as parameters. Other variables are normally documented within the source code listing.

(1) Name. CROSRF (subroutine)

Purpose:	This subroutine will produce a cross reference listing of the recipes and the menus in which they appear for those menus in the solution.
Called by:	POSTPR
Calling sequence:	CALL CROSRF
Parameters:	none
Calls:	GETMEN, LSTXRF, SORTX
Files:	Input: 26 Output: 29
Common blocks:	none
Included text:	none

(2) Name. DIBNDS (subroutine)

Purpose: This routine processes the 'BOUNDS' section of the XMP linear programing data. As each BOUNDS card is processed, DIBNDS retrieves the current bounds on the column via a call to the XMP routine XGETUB, adjusts the bounds accordingly, and transfers the revised bounds back to XMP via a call to the routine XADDUB. It should be noted that this routine is capable of processing lower bounds, upper bounds, and fixed bounds, but not 'free' variables. The reason for this restriction is that XMP is incapable (or almost so) of working with free variables. If it should become necessary to handle a free variable, the recommended response is to replace it with the difference of two non-negative variables in the problem definition.

Called by: DINPUT

Calling sequence: CALL DIBNDS(BNDTYP, IOERR, LENMA, LENMY, MAPA, MEMORY)

Input parameters:
BNDTYP - XMP bound type indicator
IOERR - file number for XMP error messages
LENMA - dimension of the mapa array
LENMY - dimension of the memory array
MAPA - XMP array
MEMORY - XMP array

Calls: DIERR, DISYM, FINDH, XADDUB, XGETUB

Files: none

Common blocks: see DIPROC

Included text: DIPROC

(3) Name. DICOLS (subroutine)

Purpose: This routine processes the 'COLUMNS' section of the XMP linear programing data. As each column is processed, DICOLS records the column name in the column name dictionary (so the column name can be converted to a column index afterward) and in the column name array (so the column index can be converted to a column name). DICOLS also sets the initial bounds on each column to be from zero to infinity and collects the non-zero coefficients in the column for a call to XADDAJ, which records the coefficients in the XMP data structures.

Called by: DINPUT

Calling sequence: CALL DICOLS (COLMAX, LENMA, LENMY, MAPA, MEMORY, BNDTYP, BIG, IOERR, COLA, COLI, MAXA, MAXN, NSTRUC)

Input parameters:
COLMAX - max number of non-zero coefficients in any column
LENMA - dimension of the MAPA array
LENMY - dimension of the MEMORY array
MAPA - XMP array
MEMORY - XMP array
BNDTYP - XMP bounds type
BIG - what XMP considers a big number
IOERR - number of XMP error file
COLA - used to hold vector of non-zero coefficients
COLI - used to hold associated vector of row numbers
MAXA - max number of non-zero coefficients
MAXN - max number of variables (columns)

Output parameters: NSTRUC - number of columns (structural variables)

Calls: DIERR, DISYM, FINDH, PUTH, STARTH, XADDAJ, XADDUB

Files: none

Common blocks: see DIPROC

Included text: DIPROC

(4) Name. DIERR (subroutine)

Purpose: The XMP data input routines use this routine to issue error messages. The calling routine identifies the error message by a number, which DIERR uses to index into a table of canned error messages. After writing the message, DIERR executes a STOP instruction.

Called by: DIBNDS, DICOLS, DINPUT, DIRHS, DIROWS

Calling sequence: CALL DIERR(IMSG)

Input parameters: IMSG - error message number

Calls: none

Files: none

Common blocks: see DIPROC

Included text: DIPROC

(5) Name. DINPUT (subroutine)

Purpose: This routine controls the reading of linear programing problem data (in the format used by the UNIVAC FMPS system) for use by XMP linear programing routines. In addition to organizing the flow of control, DINPUT enforces the rules that the input must start with a NAME card, followed by a ROWS section, a COLUMNS section, a RHS section, an optional BOUNDS section, and an ENDATA card, in that order.

Called by: XMAIN

Calling sequence: CALL DINPUT (B, BIG, BNDTYP, COLA, COLI, COLMAX, IOERR, IOIN, LENMA, LENMY, M, MAPA, MAXA, MAXM, MAXN, MEMORY, NSTRUC, ROWTYP)

Input parameters:
IOIN - input file number
MAXA - max number of non-zero coefficients
MAXM - max number of constraints
MAXN - max number of variables
COLMAX - max number of non-zero coefficients in a row
LENMA - dimension of the mapa array
LENMY - dimension of the memory array
MAPA - XMP array
MEMORY - XMP array
BNDTYP - type of bounds
BIG - what XMP considers a big number
IOERR - file number for XMP error messages
COLA - array for use as a work area
COLI - array for use as a work area

Output parameters:
M - the number of constraints
NSTRUC - the number of structural variables
ROWTYP - array of row types
B - the right-hand side

Calls: DIBNDS, DICOLS, DIERR, DIRHS, DIROWS, DISYM

Files: none

Common blocks: see DIPROC

Included text: DIPROC

(6) Name. DIPROC (included text)

Purpose: DIPROC serves to organize the common blocks used by the XMP data input routines for ease of reference. The common blocks--DIDATC, DIDATI, DDAT2C, DDAT2I, DDAT3C, and DDAT3I--are described below.

Common block DIDATC: This block contains CHARACTER type variables related to the contents of a single data card.

Variables: SYM - (Character*8) set to 'DATA' if the card is a data card. Otherwise, c contains the card type, i.e., 'ROWS', 'COLUMNS', 'BOUNDS', 'RHS', or 'ENDATA'.
CODE - (Character*2) contains the code field of the card
NAME1 - (Character*8) contains the first name field of the card
NAME2 - (Character*8) contains the second name field of the card

Common block DIDATI: This block contains non-CHARACTER type variables related to the contents of a single data card.

Variables: INFILE - (integer) the FORTRAN unit number of the input file
NCARD - (integer) the sequence number of the card being processed
VALUE - (real) the numeric value field of a data card

Common block DDAT2C: This block contains the character-type arrays used to store the row and column names in their dictionaries (one for rows, one for columns), implemented as hash tables.

Variables: RKEY - (Character*8) the row name key HASH table, dimensioned (0:198)
CKEY - (Character*8) the column name key HASH table, dimensioned (0:996)

Common block
DDAT2I: This block contains the noncharacter data used in the row and column name dictionaries.

Variables: RBIGN - (integer) the size of the row name HASH table. It is critical that the upper dimension of the RKEY and RVAL arrays be one less than RBIGN. It is also advisable for best performance, but not necessary for correct performance, that RBIGN be a prime number
RVAL - (integer) the array used to store the row indexes associated with the row names, dimensioned (0:198)
CBIGN - (integer) the size of the column name HASH table. It is critical that the upper dimension of the CKEY and CVAL arrays be one less than CBIGN. It is also advisable for best performance, but not necessary for correct performance, that CBIGN be a prime number

Common block
DDAT3C: This block contains row name and column name arrays used to convert from a row or column index to the associated name.

Variables: ROWNM - (Character*8) the row name array, dimensioned (199)
COLNM - (Character*8) the column name array, dimensioned (977)

Common block
DDAT3I: This block contains noncharacter type data associated with the row and column name arrays in DDAT3C.

Variables: NROWNM - (integer) the number of entries used in the ROWNM array
NCOLNM - (integer) the number of entries used in the COLNM array

(7) Name. DIRHS (subroutine)

Purpose: This routine processes the 'RHS' section of the XMP linear programing data. The RHS coefficient of each row is recorded in the XMP array B, with the coefficient defaulting to zero if the row does not appear in the RHS data.

Called by: DINPUT

Calling sequence: CALL DIRHS(M, B, MAXM)

Input parameters: M - the number of rows
MAXM - the dimension of the B array

Output parameters: B - the right-hand side array

Calls: DIERR, DISYM, FINDH

Files: none

Common blocks: see DIPROC

Included text: DIPROC

(8) Name. DIROWS (subroutine)

Purpose: This routine processes the 'ROWS' section of the XMP linear programing input data. Each row name is recorded in the row name dictionary via a call to PUTH, and the row type (N, G, L, or E) is converted to the appropriate XMP numeric code and stored in the XMP ROWTYP array.

Called by: DINPUT

Calling sequence: CALL DIROWS(MAXM, M, ROWTYP)

Input parameters: MAXM - max number of constraints

Output parameters: M - the number of constraints
ROWTYP - array of row types

Calls: DIERR, DISYM, FINDH, PUTH, STARTH

Files: none

Common blocks: see DIPROC

Included text: DIPROC

(9) Name. DISYM (subroutine)

Purpose: This routine reads each XMP linear programing data card. If the card is a comment card (i.e., has a '*' in column 1), it is ignored. DISYM sets the global variable SYM to 'DATA' if the card is a data card; otherwise, SYM is set to the card type ('NAME', 'ROWS', 'COLUMNS', 'RHS', 'BOUNDS', or 'ENDATA'). Data cards include a code field, two name fields, and a numeric value field; DISYM stores the contents of these fields in the global variables CODE, NAME1, NAME2, and VALUE.

Called by: DIBNDS, DICOLS, DINPUT, DIRHS, DIROWS

Calling sequence: CALL DISYM

Parameters: none

Calls: none

Files: Reads data from the FORTRAN unit number specified by INFILE, a variable in common block DIDATI.

Common blocks: see DIPROC

Included text: DIPROC

(10) Name. FINDH (subroutine)

Purpose: This routine finds a key and its associated value in a HASH table. Its use within the XMP data input routines is to convert a row or column name to its associated row or column index, once PUTH has been used to record the name and index in a HASH table. (Separate HASH tables are used for rows and columns).

Called by: XMAIN, DIBNDS, DICOLS, DIRHS, DIROWS

Calling sequence: CALL FINDH (K, V, FOUND, BIGN, TKEY, TVALUE)

Input parameters: K - the key to be searched for (defined as character*8)
BIGN - the size of the HASH table (should be a prime number for optimum performance)
TKEY - the HASH table, i.e. an array to store the keys in, dimensioned (0:bign-1)
TVALUE - an array to store the associated values in, also dimensioned (0:bign-1)

Output parameters: FOUND - .TRUE. if the item is found
V - the value associated with the key (integer)

Calls: PROBE

Files: none

Common blocks: none

Included text: none

(11) Name. GETMEN (subroutine)

Purpose: This subroutne retrieves data from the menu component file.

Called by: CROSRF, LSMNRC, SMENAT

Calling sequence: CALL GETMEN(MENNUM, NUMREC, RECNUM)

Parameters: MENNUM: Menu number (Character*10)
NUMREC: Number of recipes (integer)
RECNUM: Array of up to 30 recipe numbers (Character*10)

Calls: HASHM

Files: Input: 12
Output: 6

Common blocks: none

Included text: none

(12) Name. GETREC (subroutine)

Purpose: This subroutine retrieves recipe data from the recipe attribute file.

Called by: LSMNRC, LSTXRF

Calling sequence: CALL GETREC(RECNUM, NAME, KIND, RECRFC, RECLC, RECACC, RECNUT)

Parameters: RECNUM: Recipe number (Character*10)
NAME: Recipe name (Character*30)
KIND: Recipe kind (Character*3)
RECRFC: Recipe food cost (real)
RECLC: Recipe labor cost (real)
RECACC: Recipe acceptability (real)
RECNUT: Array of 10 nutrients (Real)

Calls: HASHR

Files: Input: 10
Output: 6

Common blocks: none

Included text: none

(13) Name. GETSOL (subroutine)

Purpose: This subroutine will read the solution from unit 8 (XMP output file) and writes two output files to be used by subsequently called subroutines. The first output file is unit 25 on which are written the deviation variables and values. The second output file is unit 26 on which are written the decision variables and values.

Called by: POSTPR

Calling sequence: CALL GETSOL(KT, IOLOG)

Parameters: KT - Counter corresponding to the number of priority levels solved by the GP algorithm
IOLOG - FORTRAN unit number corresponding to the XMP output file (unit 8)

Calls: none

Files: Input: 8 (IOLOG)
Output: 25, 26

Common blocks: none

Included text: none

(14) Name. HASHA (integer function)

Purpose: This function "hashes" a 10-character string into an integer, which is returned as the value of the function. The method used is to first divide the first eight characters of the input string into two strings of four characters each. (The final two characters of the input string do not enter into the calculation at all. However, in this application the final two characters are blanks.) These two strings are then treated as binary data, in their internal representation, and combined into a single four-character string via a bit-by-bit exclusive-or (also known as a binary add without carry). To further scramble the data, this string is split into two strings of two characters each, treated as binary integers. These integers are then multiplied together, yielding the final result.

Called by: HASHM, HASHR

Calling sequence: HASHA(NUMBER)

Input parameters: NUMBER - a recipe or menu 'number', defined as Character*10

Calls: none

Files: none

Common blocks: none

Included text: none

(15) Name. HASHM (subroutine)

Purpose: Determines the address on the menu component data file for any menu number.

Called by: GETMEN

Calling sequence: CALL HASHM(NUMBER, RBA, STOP)

Parameters: NUMBER: Menu number (Character*10)
RBA: Address on the menu component file for any recipe number (integer)
STOP: Address beyond which no searching will be done (integer)

Calls: HASHA

Files: none

Common blocks: none

Included text: none

(16) Name. HASHR (subroutine)

Purpose: Determines the address on the recipe attribute data file for any recipe number.

Called by: GETREC

Calling sequence: CALL HASHR(NUMBER, RBA, STOP)

Parameters: NUMBER: Recipe number (Character*10)
RBA: Address on the recipe attribute file for any recipe number (integer)
STOP: Address beyond which no searching will be done (integer)

Calls: HASHA

Files: none

Common blocks: none

Included text: none

(17) Name. LSGOAL (subroutine)

Purpose: This routine displays the values of the goals, deviations, and levels of achievement.

Called by: POSTPR

Calling sequence: CALL LSGOAL

Parameters: none

Calls: none

Files: Input: 18 (goals), 25 (deviation variable values)
Output: 6

Common blocks: none

Included text: none

(18) Name. LSMEN (subroutine)

Purpose: This subroutine will display the list of menus selected for inclusion in the solution along with their frequency of selection.

Called by: POSTPR

Calling sequence: CALL LSMEN

Parameters: none

Calls: none

Files: Input: 26
Output: 6

Common blocks: none

Included text: none

(19) Name. LSMNRC (subroutine)

Purpose: This subroutine will produce a listing of menus and associated recipes form an XMP solution.

Called by: POSTPR

Calling sequence: CALL LSMNRC

Parameters: none

Calls: GETMEN, GETREC

Files: Input: 8
Output: 6

Common blocks: none

Included text: none

(20) Name. LSTHDR (subroutine)

Purpose: This subroutine will display the solution report header.

Called by: XMAIN

Calling sequence: CALL LSTHDR

Parameters: none

Calls: none

Files: Output: 6

Common blocks: none

Included text: none

(21) Name. LSTOBJ (subroutine)

Purpose: This subroutine will display to the user the menu attribute associated with the current objective in the goal programing algorithm. This results in a display of the priority ordering associated with a particular solution.

Called by: XMAIN

Calling sequence: CALL LSTOBJ(OBJROW)

Parameters: OBJROW - Name of current objective row (Character*8)

Calls: none

Files: Output 6

Common blocks: none

Included text: none

(22) Name. LSTXRF (subroutine)

Purpose: This subroutine will display the cross reference list for those menus that were selected for inclusion in the solution. The total number of times that each recipe is to be served during the menu cycle will also be displayed.

Called by: CROSRF

Calling sequence: CALL LSTXRF

Parameters: none

Calls: GETREC

Files: Input: 29
Output: 6

Common blocks: none

Included text: none

(23) Name. MEMCON (common block)

Purpose: This common block was inserted in several of the XMP subroutines in order to overcome the argument checking errors generated by the Burroughs FORTRAN 77 compiles. It contains the integer fields of the real MEMORY array.

Variables: IMEM = corresponds to the main storage array (MEMORY) and is held in common for the purpose of addressing integer portions of that array. The dimension of IMEM is equal to LENMY, the same as MEMORY, but is listed as a constant; therefore if LENMY changes, each occurrence of IMEM must be redimensioned.

Used in: XMAIN, XADDAJ, XADDUB, XBTRAN, XFACT, XFEAS, XFTRAN, XPHAS2, XUPDAT

(24) Name. POSTPR (subroutine)

Purpose: This subroutine is called by the solution algorithm after completing the solution to the final priority. This subroutine then calls the subroutines which will read the final solution and display the following five reports:

- o List of selected menus
- o Summary of the menu attributes for selected menus
- o Menu planning goals and levels of achievement
- o List of selected menus with associated recipe names
- o Cross reference list of recipes and menus in which they appear

Called by: XMAIN

Calling sequence: CALL POSTPR(IOLOG)

Parameters: IOLOG - FORTRAN unit number for XMP output

Calls: CROSRF, GETSOL, LSGOAL, LSMEN, LSMNRC, SMENAT, SUMMRY, TRMCOD

Files: none

Common blocks: none

Included text: none

(25) Name. PROBE (subroutine)

Purpose: This routine probes a HASH table for a 'key' (a character string). As a result of the probe, the user of the subroutine is told (1) whether the key is in the table; (2) if so, its location; (3) if not, the location where it should go (unless the table is full); and (4) whether the table is full. The HASH table is organized in the following manner: Given an eight-character key, the key is split into two four-character strings which are treated as binary data and crunched into a single four-character string via the exclusive-or function (also known as a binary add without carry). The resulting string is then treated as a binary integer and reduced module the size of the HASH table, yielding an integer in the range from 0 to BIGN-1, where BIGN is the size of the HASH table. This integer is used for the initial probe into the table. In the case of a collision (i.e., when the slot for the initial probe is already occupied with another key), quadratic probing is used to continue the process, i.e., successive probes are at H+1, H+4, H+9, etc., where H is the initial probe. To the reader not familiar with HASH table techniques, this may seem to be a lot of uneccessary rigamarole; however, the point of all this machination is that keys can be located very rapidly on the average (faster than by binary search, for example), and the programing is not all that complex. Several computer science texts describe HASH tables, among them Algorithms + Data Structures = Programs, by N. Wirth.

Called by: FINDH, PUTH

Calling sequence: CALL PROBE (K, FOUND, FULL, H, BIGN, TKEY)

Input parameters: K - the key to be located (defined as Character*8)
BIGN - the size of the HASH table (this number should be a prime number for optimum performance)
TKEY - the HASH table, i.e., an array to store the keys in, dimensioned (0:bign-1)

Output parameters: FOUND - .TRUE. if the key is found in the table. In this case, H = the index of the key. If the key is not found, then H = the index where it should be added.
FULL - .TRUE. if the table is full (i.e., this key cannot be added)

Calls: none

Files: none

Common blocks: none

Included text: none

(26) Name. PUTH (subroutine)

Purpose: This routine records a character string 'key' and an associated integer value in a HASH table. Its use within the XMP input data routines is to record row or column names and their associated row or column indicies. Once a key has been recorded, its associated index may be retrieved via a call to FINDH.

Called by: DICOLS, DIROWS

Calling sequence: CALL PUTH (K, V, FULL, BIGN, TKEY, TVALUE)

Input parameters: K - the key of this item (defined as Character*8)
V - the associated value (integer)
BIGN - the size of the HASH table (should be a prime number for optimum performance)
TKEY - the HASH table, i.e. an array to store the keys in, dimensioned (0:bign-1)
TVALUE - an array to store the associated values in, also dimensioned (0:bign-1).

Output parameters: FULL - .TRUE. if the HASH table is full, i.e., this item can't be added

Calls: PROBE

Files: none

Common blocks: none

Included text: none

(27) Name. SMENAT (subroutine)

Purpose: This subroutine will read the solution of selected menus from unit 26, and write two files for use by subsequently called subroutines. The first of these files is written to unit 27 and consists of a listing of the menus and their associated recipes. The second file is written to unit 28 and consists of the menus and their associated attributes.

Called by: POSTPR

Calling sequence: CALL SMENAT

Parameters: none

Calls: GETMEN

Files: Input: 14, 26
Output: 27, 28

Common blocks: none

Included text: none

(28) Name. STARTH (subroutine)

Purpose: This routine initializes a hash table to its initial 'empty' state. STARTH must be called before any other use of the hash table, i.e., calls to PUTH, FINDH, or PROBE.

Called by: DICOLS, DIROWS

Calling sequence: CALL STARTH (BIGN, TKEY)

Input parameters: BIGN - the size of the hash table
TKEY - the array of keys, defined as character*8 and dimensioned (0:bign-1)

Calls: none

Files: none

Common blocks: none

Included text: none

(29) Name. SUMMRY (subroutine)

Purpose: This subroutine will display a summary of a portion of the menu attribute file for those menus selected for inclusion in the solution.

Called by: POSTPR

Calling sequence: CALL SUMMRY

Parameters: none

Calls: none

Files: Input: 28
Output: 6

Common blocks: none

Included text: none

(30) Name. TRMCOD (subroutine)

Purpose: This subroutine will count the termination codes in the solution file and check for infeasible or unbounded solutions.

Called by: POSTPR

Calling sequence: CALL TRMCOD(KOUNT, IERRT, IOLOG)

Parameters: KOUNT - Counter for feasible terminations
IERRT - Error flag
IOLOG - FORTRAN file number for XMP output

Calls: none

Files: Input: 8
Output: 6

Common blocks: none

Included text: none

(31) Name. XGOAL (subroutine)

Purpose: This routine performs one iteration of the sequential linear goal programing algorithm. XGOAL accomplishes this in four steps. First, the routine changes the objective function to the new objective row. Second, it changes the old objective row, and a list of rows associated with the new objective function, from nonconstraining to equality type constraints. Third, it sets the right hand side of the old objective row equal to the optimal objective function value from the previous linear programing problem. And finally, it invokes XMP to solve the new linear programing problem, starting with the existing basis from the previous problem.

Called by: XMAIN

Calling sequence: CALL XGOAL (B, BASCB, BASIS, BASLB, BASUB, BLOW, BNDTYP, BOUND, CAND, CANDA, CANDCJ, CANDI, CANDL, COLA, COLI, COLMAX, FACTOR, IEROW, IOERR, IOLOG, ITER1, ITER2, LENMA, LENMI, LENMY, LEROW, LOOK, M, MAPA, MAPI, MAXM, MAXN, MEMORY, MINMAX, N, NEROW, NEWOBJ, NSTRUC, NTYPE2, OLDOBJ, PICK, PRINT, ROWTYP, STATUS, TERMIN, UNBDDQ, UZERO, XBZERO, YQ, Z)

Input parameters:
B - the right-hand side array
BLOW - contains the lower limit for each two-sided constraint
BNDTYP - bound type indicator
BOUND - is the common upper bound when BNDTYP=2
COLMAX - the max number of non-zeros in any matrix column
FACTOR - the refactorization frequency
IEROW - an array of indexes of rows which should be changed to equality constraints
IOERR - the IO unit for error messages
IOLOG - the IO unit for log information
LENMA - the length of the MAPA array
LENMA -the length of the MAPI array
LENMY - the length of the MEMORY array
LEROW - the length of the IEROW array
LOOK - the number of columns to be considered during construction of the candidate list
M - the number of constraints
MAPA - XMP data map array
MAPI - XMP data map array
MAXM - the maximum number of constraints
MAXN - the maximum number of variables
MEMORY - the main storage array

MINMAX - +1 to maximize, -1 to minimize
NEROW - the number of entries in the IEROW array
NEWOBJ - the index of the row for the new objective function
NSTRUC - the number of structural variables
NTYPE2 - number of upper bounded constraints, if bndtyp=2
OLDOBJ - the index of the row of the old objective function
PICK - the size of the candidate list
PRINT - specifies the level of printing desired
ROWTYP - array of row types
Z - the value of the old objective function

Output parameters: ITER1 - number of phase 1 SIMPLEX iterations
ITER2 - number of phase 2 SIMPLEX iterations
N - the total number of variables (including slacks, artificials)
STATUS - array of variable statuses
TERMIN - SIMPLEX algorithm termination code
UNBDDQ - if the problem is unbounded, the index of the variable about to enter the basis
UZERO - array of dual variable values
XBZERO - array of basic variable values
Z - the value of the objective function

Parameters that are used as work areas:
Arrays: BASCB, BASIS, BASLB, BASUB, CAND, CANDA, CANDCJ, CANDI, CANDL, COLA, COLI

Calls: XADDUB, XPRIML, XSETC, XSTART

Files: none

Common blocks: none

Included text: none

(33) Name. XMAIN (main program)

Purpose: This is the main program for the solution algorithm. It controls the logical flow of operations and initializes parameters. This program was modified from the original XMP demonstration program to incorporate an algorithm for sequential linear goal programing. The programer should refer to Appendix B for a complete description of variables.

Called by: Not applicable.

Calling sequence: Not applicable.

Parameters: none

Calls: DINPUT, FINDH, LSTHDR, LSTOBJ, POSTPR, XGOAL, XMAPS, XPRIML, XPRINT, XSETC, XSLACK

Files: Input: 5, 19
Output: 8

Common blocks: see DIPROC

Included text: DIPROC

(33) Name. XPRINT (subroutine)

Purpose: This routine prints out the current basic solution and the objective function value for the XMP linear programing problem. XPRINT is one of the original XMP routines written by Roy E. Marsten of the University of Arizona, but has been modified to print column names as they appear in the input data rather than XMP variable numbers.

Called by: XMAIN, XPHASE2

Calling sequence: CALL XPRINT(BASIS,BNDTYP,BOUND, IOERR,IOLOG, LENMA,LENMY,M,MAPA,MAXM,MAXN,MEMORY,N,NTYPE2, STATUS,XBZERO,Z)

Input parameters: BASIS is the list of basic variables
BNDTYP - 1 means lower bound=0, upper bound = +infinity for every non-free variable; 2 means lower bound = 0, upper bound = bound for every non-free structural variable; 3 means lower bound = 0 for every non-free variable; 4 means both bounds are general.
BOUND is the common upper bound when BNDTYP = 2.
IOERR is the i/o unit where error messages are to be written.
IOLOG is the i/o unit where log information is to be written, if requested.
LENMA is the length of the MAPA array.
LENMY is the length of the MEMORY array.
M is the number of constraints.
MAPA is a map of the data structure for the original problem data, consisting of pointers into the memory array.
MAXM is the maximum number of constraints that will be encountered during the current run; it is used to set array sizes.

MAXN is the maximum number of variables that will be encountered during the current run; it is used to set array sizes.
MEMORY is the main storage array; it contains all of the arrays for the problem data and the basis inverse.
N is the number of variables (total, including slacks and artificials).
NTYPE2: if BNDTYP=2, then each variable 1, 2, ..., NTYPE2 must either share the common upper bound or else be a free variable; each of the remaining variables NTYPE2+1, ... , n must either have lower bound zero and upper bound +infinity or else be a free variable.
STATUS: an indicator for each variable: 0 means the variable is out at its lower bound; k means that this is the k-th basic variable; -1 means that this variable is out at its upper bound; -2 means that this is a free variable; once in the basis it never leaves; -3 means that this is an artificial variable; once it leaves the basis it never re-enters; -4 means the variable is locked out of the basis at its lower bound; -5 means the variable is locked out of the basis at its upper bound. NOTE: free and artificial variables always have status -2 or -3 respectively, they do not get a positive status when they are in the basis. NOTE: super-basic variables have positive status greater than m.
XBZERO is the array containing the values of the basic variables and the super-basic variables.
Z is the value of the objective function.

Calls: XGETUB

Files: none

Common blocks: DIDAT3 (see DIPROC)

Included text: none

(34) Name. XSETC (subroutine)

Purpose: This routine sets the objective function of the XMP linear programing problem equal to a (nonconstraining) row of the LP problem matrix. This is done by retrieving each column of the matrix in turn and locating the coefficient of the specified row. (It may be that the row is not found in the list of coefficients for a column, which means that the coefficient in this column is zero.) XSETC then calls XSETCJ to set the objective coefficient for that column.

Called by: XMAIN, XGOAL

Calling sequence: CALL XSETC (COLA, COLI, COLMAX, IROW, IOERR, +LENMA, LENMY, MAPA, MAXN, MEMORY, MINMAX, NSTRUC)

Input parameters:
COLMAX - the maximum number of non-zero coefficients in a column (and the length of the COLA and COLI arrays)
IROW - the index of the row to become the objective function
IOERR - IO unit for error messages
LENMA - length of the MAPA array
LENMY - length of the MEMORY array
MAPA - XMP map array
MAXN - the maximum number of variables
MEMORY - the main XMP array
MINMAX - +1 if maximizing, -1 if minimizing
NSTRUC - the number of structural variables

Parameters used as work areas:
Arrays: COLA and COLI

Calls: XGETAJ, XSETCJ

Files: none

Common blocks: none

Included text: none

(35) Name. XSETCJ (subroutine)

Purpose: Given a column of the XMP linear programing problem matrix, this routine sets the objective function coefficient in that column equal to a value specified by the user.

Called by: XSETC

Calling sequence: CALL XSETCJ (CJ, IOERR, J, MAXN, PROFIT)

Input parameters:
CJ - the new objective function coefficient
IOERR - io unit for error messages
J - the index of the column involved
MAXN - length of the profit array
PROFIT - the array holding the objective function coefficients

Calls: none

Files: none

Common blocks: none

Included text: none

3-5. SUMMARY. The information presented in this chapter was intended to give the knowledgeable programer the necessary information to maintain, and ultimately, modify the model. The information in this chapter should be used in conjunction with the discussion of the model structure presented in the previous chapter and the documented source code listings.

APPENDIX A

STUDY CONTRIBUTORS

1. STUDY TEAM

   a. Study Director

      CPT(P) August C. Manguso, Analysis Support Directorate

   b. Team Members

      Mr. John Warren, OR Analyst
      Dr. Dong Kim, Mathematician

   c. Other Contributors

      Ms Pat Fleming

2. EXTERNAL CONTRIBUTORS

      SFC William J. Morris, US Army Logistics Management Center,
      FT Lee, VA

APPENDIX B

INTRODUCTION TO XMP

The following XMP documentation is reprinted in its entirety by permission of the author, Professor Roy E. Marsten.

Date last modified: July 20, 1981

XMP is a structured library of subroutines for experimental mathematical programming. Development of the original version of the XMP library was supported by the National Science Foundation under grants MCS76-01311 and MSC76-01311 A01 (1976-1978) to the Center for Computational Research in Economics and Management Science at the Massachusetts Institute of Technology. (At the time of the initial grant the Center was part of the National Bureau for Economic Research, Inc.) The principal investigator was Roy Marsten.

XMP is now being distributed to universities and government agencies by the Department of Management Information Systems at the University of Arizona and to corporations by the XMP Optimization Software Co.

A thorough introduction to XMP is contained in: The Design of the XMP Library, Transactions on Mathematical Software, to appear December 1981.

Inquiries concerning XMP should be directed to:

Prof. Roy E. Marsten
Department of Management Information Systems
College of Business and Public Administration
University of Arizona
Tucson, Arizona 85721

Phone: (602) 626-3116

The current version of XMP uses the LA05 routines from the Harwell Library to manage an LU factorization of the basis matrix. The LA05 routines were written by John K. Reid. Inquiries concerning the Harwell Library should be directed to:

Computer Science and Systems Division
AERE Harwell
Oxfordshire, England

Reference: FORTRAN Subroutines for Handling Sparse Linear Programming Bases, John K. Reid, Report AERE-R8269, January 1976.

The standard XMP tape contains three files.

File 1	System documentation and the data for a small test problem in the form that can be read by the sample user program.
File 2	Three different versions of the XMAPS routine and three different versions of the six Harwell routines (LA05A, LA05B, LA05C, LA05E, MC20A, MC20B). The three versions are suitable for DEC, IBM, and CDC computers. These three versions have sufficed for all of the computers that have been encountered so far.
File 3	The sample user program and the 39 routines that make up the standard XMP library.

To use the XMP library on your computer you must select one of the three standard versions: DEC, IBM, or CDC. These versions differ in the types of variable and array declarations that they use. These are:

DEC	Double precision Real Integer
IBM	Double precision Real Integer Integer*2
CDC	Real Integer

(For example, use the CDC version on a Burroughs computer, the DEC version on a UNIVAC computer.)

Copy the desired version of the routines in File 2, and then edit File 3 as follows.

DEC	No editing is required
IBM	Locate each of the 31 occurrences of the line: The next statement should specify half-words if possible.

Immediately following each of these lines is an integer declaration that should be changed to INTEGER*2.
NOTE: You may use the DEC version on an IBM computer but switching to INTEGER*2 array declarations will save considerable main memory space for large problems.

CDC — Make the following global substitutions:

From	To
DOUBLE PRECISION	REAL
DABS	ABS
D1	E1
LA05AD	LA05A
LA05BD	LA05B
LA05CD	LA05C

NOTE: The D1 to E1 substitution is for format codes. The double precision versions of the LA05 routines have a D appended to their director of the subroutines in the library. Date last modified: June 8, 1981.

There are six categories of subroutines in the XMP Library:

1) Subroutines that implement the logic of the SIMPLEX method.

2) Subroutines that serve as an interface between the SIMPLEX method and the data structure for the problem data.

3) Subroutines that serve as an interface between the SIMPLEX method and the data structure for the basis inverse representation.

4) Subroutines that manage the data structure for the problem data.

5) Subroutines that manage the data structure for the basis inverse representation.

6) Subroutines that provide miscellaneous support services.

Category 1. Subroutines that implement the logic of the SIMPLEX method.

XBCOMP	Computes the current values of the basic variables and the current value of the objective function.
XBREDU	Reduces the right-hand-side to account for the non-basic variables which are at non-zero bounds.
XCAND	Constructs the candidate list. This is the list of attractive non-basic variables that are eligible to enter the basis during the subsequent series of minor iterations.

XCHECK	Checks the accuracy of the current primal and dual solutions.
XCHURZR	Determines the variable to leave the basis for a primal SIMPLEX pivot.
XDCHQ	Determines the variable to enter the basis for a dual SIMPLEX pivot.
XDCHR	Determines the variable to leave the basis for a dual SIMPLEX pivot.
XDOT	Computes the inner product between a row vector and a packed matrix column.
XDPH2	Executes Phase 2 for the dual SIMPLEX method.
XDUAL	Top level routine for the dual SIMPLEX method.
XFEAS	Starts from any given basis and finds a primal feasible basis, if one exists. This routine is used as a Phase 1 for the primal SIMPLEX method.
XPHAS2	Executes Phase 2 of the primal SIMPLEX method.
XPIVOT	Pivots the chosen entering variable into the basis (primal SIMPLEX method).
XPRIML	Top level routine for the primal SIMPLEX method.
XPTIE	Resolves ties that arise during the ratio test for the primal SIMPLEX method (XCHUZR).
XSLACK	Sets up a starting basis with a slack variable for each less-than-or-equal constraint, a surplus variable for each greater-than-or-equal constraint, an artificial variable (with both bounds zero) for each equation, and a free variable for each free row. To be used with XPRIML (which call XFEAS) or XDUAL.
XSTART	Used to start the primal or dual SIMPLEX method from any given basis.
XUPDX	Updates the primal solution and the objective function value.
XZCOMP	Computes the current value of the objective function.

FCAND	The fast version of XCAND: accesses the problem data structure directly.
FDCHQ	The fast version of XDCHQ: accesses the problem data structure directly.

Category 2. Subroutines that serve as an interface between the SIMPLEX method and the data structure for the problem data.

XADDAJ	Adds a single column to the existing linear program (calls XDATA1).
XADDUB	Adds bounds for a single variable (calls XDATA 3).
XGETAJ	Gets a single column from the existing linear program (calls XDATA2).
XGETUB	Gets the bounds for a single variable (calls XDATA4).
Category 3	Subroutines that serve as an interface between the SIMPLEX method and the data structure for the basis inverse representation.
XBTRAN	Performs the "backward transformation," i.e., row vector * basis inverse (calls LA05B).
XFACT	Re-factors (or re-inverts) the current basis (calls LA05A).
XFTRAN	Performs the "forward transformation," i.e., basis inverse * column vector (calls LA05B).
XUPDAT	Updates the current factorization (or inverse) of the basis (calls LA05C).
Category 4	Subroutines that manage the data structure for the problem data.

XDATA1	Adds a single column to the problem data structure (called by XADDAJ).
XDATA2	Retrieves a single column from the problem data structure (called by XGETAJ).
XDATA3	Adds bounds for a single variable to the problem data structure (called by XADDUB).
XDATA4	Retrieves the bounds for a single variable from the problem data structure (called by XGETUB).

Category 5. Subroutines that manage the basis inverse representation.

NOTE: In this version of XMP the basis inverse is managed by the LA05 routines written by John K. Reid at Harwell.

LA05A	Factors the basis into L and U factors (called by XFACT).
LA05B	Performs both the "backward transformation" and "forward transformation" operations (called by XBTRAN and XFTRAN).
LA05C	Updates the factorization of the basis after a column exchange (called by XUPDAT).
/LA05D/	A labelled common area for numerical constants.
LA05E	A list compressor (called by LA05A and LA05C).
MC20A	A sorting program (called by LA05A).
XLA05X	An extra routine for interfacing the LA05A routine with XMP (called by XFACT).

Category 6. Subroutines that provide miscellaneous support services.

XCONSA	Sets constants in the data structure for the problem data (called by XMAPS).
XCONSI	Sets constants in the data structure for the basis inverse representation (called by XMAPS).
XLOG	Prints log information after each pivot, if requested.
XMAPS	Sets up the map of the data structure for the problem data (MAPA) and the map of the data structure for the basis inverse representation. (NOTE: this must be the first XMP routine called by the user program.)
/XMPCOM/	A labelled common area for numerical constants.
XPRINT	Prints the current basic solution and objective function value.
XSTOP	Provides for centralized handling of all fatal errors.

WRITING A USER PROGRAM

To use XMP you must write a driver program of your own. Refer to the sample user program, XDEMO (this is XMAIN in the menu planning model), while reading these instructions. XDEMO may be modified and expanded to meet your particular needs.

The user program must do the following:

1) Set MAXM, MAXN, MAXA, COLMAX, PICK, and LENMY to suitable values. These values fix the sizes of the necessary arrays. The setting of LENMY will be explained in the DETERMINING MAIN MEMORY REQUIREMENTS section below.

MAXM is the maximum number of constraints that will be encountered during the current run.

MAXN is the maximum number of variables that will be encountered during the current run.

MAXA is the maximum number of non-zeros that will be encountered in the whole constraint matrix during the current run.

COLMAX is the maximum number of non-zeros that will be encountered in any single matrix column during the current run.

PICK is the size of the candidate list used for multiple printing.

LENMY is the length of the memory array.

MEMORY is the main storage array . . . it contains all of the arrays for the problem data and the basis inverse representation.

When setting MAXN and MAXA, be sure to allow the logical (slack, surplus, artificial, or free) variable that will be generated by XSLACK for each constraint. That means MAXM extra variables and MAXM extra non-zeros. Set PICK to the largest value that will be used during the current run.

2) Declare and dimension all of the necessary XMP arrays. The actual dimensions used must correspond to the settings in 1) above.

See the XMP dictionary for the definitions of these arrays.

Double precision arrays (or real, as appropriate):

Array	Number of entries
B	MAXM
BASCB	MAXM
BASLB	MAXM
BASUB	MAXM
BETAR	MAXM (needed for dual SIMPLEX method)
BLOW	MAXM (needed for two-sided constraints)
CANDA	COLMAX, PICK
CANDCJ	PICK
COLA	COLMAX
MEMORY	LENMY
UZERO	MAXM
XBZERO	MAXM
YQ	MAXM

Integer arrays:

Array	Number of entries
MAPA	20
MAPI	20

INTEGER*2 arrays (or INTEGER, as appropriate):

Array	Number of entries
BASIS	MAXM
CAND	PICK
CANDI	COLMAX, PICK
CANDL	PICK
COLI	COLMAX
ROWTYP	MAXM
STATUS	MAXN

3) Set IOERR, IOLOG, PRINT, FACTOR, and LOOK to appropriate values (these values may be changed at any time desired).

IOERR is the I/O unit where error messages are to be written.

IOLOG is the I/O unit where log information is to be written, if requested.

PRINT specifies the level of printing desired:

1 means termination condition messages (optimal solution, unbounded, infeasible);

2 means print the objective function value after every basis refactorization;

3 means log information at every iteration.

FACTOR LOOK is the number of matrix columns to be scanned during construction of the candidate list . . . controls partial pricing.

4) Call XMAPS to set the pointers in MAPA and MAPI. These are the maps of the problem data structure and the basis inverse data structure, respectively.

5) You are now ready to use the remaining XMP routines. A typical sequence, as in XDEMO, would be:

XADDAJ - to add the columns of the linear program,

XADDUB - to add upper and lower bounds on the variables,

XSLACK - to set up a starting basis,

XPRIML - to solve the LP by the primal SIMPLEX method,

XPRINT - to print out the solution.

NOTE: The convention in XMP is to maximize the objective function. If you want to minimize CX, the fact that MIN CX = - MAX (-C)X, that is, enter the negative of each objective function coefficient. Then, after the optimal solution has been found, just take the negative of the objective function value and the negative of each dual variable (see XDEMO or XMAIN in menu planning model).

NOTE: The execution speed of XMP for any given problem will be strongly affected by the values you choose by LENMY, LOOK, and PICK. LENMY determines how much space is available for the factors of the basis. If LENMY is too small, then the basis factors will have to be compressed too often, which slows down execution. LENMY should be chosen to give a growth factor (computed and printed out by XMAPS) of about 7.0. LOOK is the number of columns priced out in order to set up a candidate list of attractive non-basic variables. LOOK should be between 50 and 150, depending on the size of the problem. PICK is the number of attractive non-basics that are placed in the candidate list. PICK should be between 3 and 8 for most applications. Execution speed and stability are also effected by the value you choose for FACTOR, the re-factorization fequency. FACTOR = 50 is reasonable for most problems.

NOTE: Individual matrix columns are passed to XMP through the XADDAJ routine. What is actually passed is the number of non-zero entries, a list of the non-zero entries, and a list of the corresponding row numbers. The non-zeros do not have to be in order, i.e., the row numbers do not have to be increasing.

NOTE: XSLACK will add one logical (slack, surplus, artificial, or free) variable for each constraint. You do not have to provide these logical variables yourself. You do have to set the ROWTYP and STATUS arrays before calling XSLACK. The right-hand-side elements do not have to be non-negative.

NOTE: If your model contains two-sided contraints (ROWTYP = 2) (BLOW < = AX < = B), then you must provide an extra array, BLOW(MAXM), which is dimensioned the same as the usual right-hand-side, B(MAXM). You must also use BNDTYP = 3 or 4.

NOTE: If your model contains free variables (lower bound = -infinity, upper bound = +infinity), other than those introduced by XSLACK for free rows (ROWTYP = 2), then you must pivot them into the basis yourself. You may use XPIVOT to do this. A safer procedure, however, would be to replace each of your free variables by a difference of two non-negative variables (see any standard LP text).

DETERMINING MAIN MEMORY REQUIREMENTS

On a control data machine, where everything is in terms of whole words, the amount of array storage used is given by:

CDCLEN = LENMY + 9*MAXM + MAXN + 2*COLMAX + 2*COLMAX*PICK + 3*PICK + 40

In words, where LENMY is expressed in words (not including BETAR or BLOW).

On an IBM machine with double words and half words as well as whole words available, the amount of storage used is given by:

IBMLEN = 8*LENMY + 68*MAXM + 2*MAXM + 10*COLMAX + 10*COLMAX*PICK + 12*PICK + 160

In bytes, where LENMY is expressed in double words (not including BETAR or BLOW)

On a machine such as the DEC, with double words or single words for real numbers but without INTEGER*2, we have:

DECLEN = (1/2)*LENMY + 16*MAXM + MAXN + 3*COLMAX + 3*COLMAX*PICK + 4*PICK + 40

In words, where LENMY is expressed in double words (not including BETAR or BLOW)

Setting LENMY: The MEMORY array will hold the data structure for the problem data and the data structure for the basis inverse representation. In the current version of XMP, these data structures are as follows:

Problem data

Array	Type	No. of entries
LOWERB	REAL	MAXN if BNDTYP = 4; 0 otherwise
UPPERB	REAL	MAXN if BNDTYP = 3 or 4; 0 otherwise
PROFIT	REAL	MAXN
ACOEFF	REAL	MAXA
COLPNT	INTEGER	MAXN + 1
ROWNOS	INTEGER*2	MAXA
MAXA	INTEGER	1
MAXN	INTEGER	1
MAXM	INTEGER	1

Basis inverse (J. K. Reid)

Array	Type	No. of entries
G	DOUBLE PRECISION	1
U	DOUBLE PRECISION	1
A	DOUBLE PRECISION	1A
W	DOUBLE PRECISION	MAXM
IP	INTEGER	MAXM,2
IND	INTEGER*2	IA,2
IW	INTEGER*2	MAXM,8
IA	INTEGER	1

(DOUBLE PRECISION is replaced by REAL and INTEGER*2 by INTEGER, as appropriate.)

Note that the original problem data (LOWERB, UPPERB, PROFIT, and ACOEFF) are held in single precision even if the computations are to be done in double precision. This is almost always adequate.

Reid uses the A and IND arrays to hold the factors of the current basis. The maximum amount of space that will be needed for these factors is unpredictable. The user should make LENMY as large as possible. The XMAPS routine will compute the amount of space required for all of the fixed length arrays and then set IA so as to allocate all of the remaining space to the A and IND arrays. As a rule of thumb, LENMY should be large enough so that we get

IA > = 4*DENSE*MAXM**2

where DENSE = MAXA / (MAXM*MAXN) is an estimate of the density of the average basis.

Therefore, on a Control Data machine, we should take (assuming BNDTYP = 1 or 2):

LENMY > = 11*MAXM + 2*MAXN + 2 MAXA + 12 DENSE*MAXM**2

where LENMY is in words (add MAXN words if BNDTYP = 3 or 2*MAXN words if BNDTYP = 4).

On an IBM machine we should take (assuming BNDTYP = 1 or 2):

LENMY > = MAXN + (3/4)*MAXA + 4*MAXM + (3/2)*(4*DENSE*MAXM**2)

where LENMY is in double words. (NOTE: in the IBM version XMAPS rounds MAXA, MAXM, nad MAXN up, if necessary, so that they are multiples of 4.) (Add MAXN/2 double words if BNDTYPE = 3 or MAXN double words if BNDTYP = 4.)

For the DEC we should take (assuming BNDTYP = 1 or 2):

LENMY > = 6*MAX + (3/2)*MAXN + MAXA + 8*DENSE*MAXM**2

where LENMY is in double words. (Add MAX/2 double words if BNDTYPE = 3 or MAX double words if BNDTYP = 4.)

Rather than using the formula for LENMY, you may just guess a value. The XMAPS routine will compute and print out a growth factor, which estimates how much growth in the basis factors there is room for, given the specified value of LENMY. If this growth factor is less than 3.0, then the run will be terminated and you may increase LENMY and try again. If the growth factor is greater than 15.0, then LENMY is probably unnecessarily large and may be reduced. A growth factor of about 7.0 is usually about right.

Dictionary of Variable and Array Names used in XMP
(date last modified: June 8, 1981)

ACOEFF	is part of the problem data structure. It contains the non-zero coefficients
B	is the right-hand-side array
BASCB	is an array containing the objective function coefficients of the basic variables
BASIS	is the list of basic variables
BASLB	is an array containing the lower bounds on the basic variables
BASUB	is an array containing the upper bounds on the basic variables
BETAR	is used to hold row R of the basis inverse where R is the position of the variable BASIS(R) that is leaving the basis
BLOW	contains the lower limit for each two-sided contraint (upper limit is in R)

BNDTYP	1 means lower bound = 0, upper bound = +infinity for every non-free variable; 2 means lower bound = 0, upper bound = BOUND for every non-free structural variable; 3 means lower bound = 0 for every non-free variable 4 means both bounds are general
BOUND	is the common upper bound when BNDTYP = 2
CAND	is the candidate list; it contains the non-basic variables that are eligible to enter the basis during a series of minor iterations
CANDA	is a table containing the non-zero coefficients of the columns that belong to the candidate list
CANDCJ	is a list containing the objective coefficient for each column that belongs to the candidate list
CANDI	is a table containing the two numbers corresponding to the non-zeros of the columns that belong to the candidate list
CANDL	is a list containing the number of non-zeros in each column that belongs to the candidate list
CJ	is used to hold one objective function coefficient
COLA	is used to hold the non-zero coefficients of a matrix problem
COLI	is used to hold the row numbers corresponding to the non-zeros of a matrix column
COLLEN	is used to hold the number of non-zeros in a matrix column
COLMAX	is the maximum number of non-zeros in any matrix column
COLPNT	is part of the problem data structure. It contains a pointer to the beginning of the section for each column in the ACOEFF and ROWNOS arrays
DERROR	Output by XCHECK: the absolute value of the maximum dual residual

DFEASQ If a dual infeasible column is detected, during the dual SIMPLEX method, then DFEASQ is the index of the corresponding primal variable

DINDEX Output by XCHECK: the index of the column where the maximum dual residual occurs

DOK Output by XCHECK: .TRUE. means current solution passes the dual accuracy test; .FALSE. means it fails

DQ is the relative profit or reduced profit of the entering variable

DTERM is the termination code for the dual SIMPLEX method:
1 means optimal solution found;
2 means the dual is unbounded;
3 means dual feasibility lost;
4 means the presumed optimal solution does not satisfy the accuracy check;
5 means that it was not possible to make a dual feasible start

DTOLER Input for XCHECK: the tolerance for the dual accuracy check

DUNBR If the dual problem is unbounded, then DUNBR is the row in which the unbounded condition was detected

FACTIT is the number of iterations performed since the last factorization

FACTOR is the re-factorization frequency

FCODE is a return code for XFACT:
1 means everything OK;
2 means some artificial variables have been inserted into the basis;
3 means storage overflow

FEAS .TRUE. if a feasible solution of the original problem has been found; .FALSE. otherwise

INTYP +1 means the entering variable is increasing from its lower bound
-1 means the entering variable is decreasing from its upper bound

IOERR	is the I/O unit where error messages are to be written
IOLOG	is the I/O unit where log information is to be written, if requested
J	is used to hold the index of a single variable
LAMBDA	is the winning ratio from the dual ratio test
LEAVE	is the index of the variable that leaves the basis when variable Q enters, LEAVE = Q if OUTTYP = 0, otherwise LEAVE = BASIS(R)
LENMA	is the length of the MAPA array
LENMI	is the length of the MAPI array
LENMY	is the length of the MEMORY array
LOOK	is the number of matrix columns to be considered during construction of the candidate list
LOWERB	is part of the problem data structure. It contains the lower bounds on the variables when BNDTYP = 4
LJ	is used to hold the lower bound on a single variable
LQ	is the lower bound on the entering variable
M	is the number of constraints
MAPA	is a map of the data structure for the original problem data, consisting of pointers into the memory array
MAPI	is a map of the data structure for the basis inverse representation, consisting of pointers into the memory array
MAXA	is the maximum number of non-zeros that will be encountered in the whole constraint matrix during the current run

MAXM is the maximum number of constraints that will be encountered during the current run; it is used to set array sizes

MAXN is the maximum number of variables that will be encountered during the current run; it is used to set array sizes

MEMORY is the main storage array; it contains all of the arrays for the problem data and the basis inverse representation

N is the number of variables (total, including slacks and artificials)

NTYPE2 If BNDTYP = 2, then each variable 1,2,...NTYPE2 must either share the common upper bound or else be a free variable. Each of the remaining variables NTYPE2+1,...N must either have lower bound zero and upper bound +infinity or else be a free variable

NREJ The number of entries in the reject list sent to XDCHR (see reject definition)

OUTTYP +1 means the leaving variable is going to its lower bound;
-1 means the leaving variable is going to its upper bound;
0 means the entering and leaving variables are the same

PCODE A return code for XPIVOT. PCODE = -1 if the PIVOT column has been rejected because the PIVOT element is too small; otherwise PCODE is set equal to the value of UCODE returned by XUPDAT

PERROR Output by XCHECK: the absolute value of the maximum primal residual

PHASE is the phase, 1 or 2

PICK is the size of the candidate list used for multiple pricing

PINDEX Output by XCHECK: the index of the row where the maximum primal residual occurs

PIVOT is the pivot element, PIVOT = YQ(R)

POK Output by XCHECK: .TRUE. means the current solution passes the primal accuracy test; .FALSE. means it fails

PPRIME is the number of attractive non-basic variables actually placed in the candidate list; PPRIME less than or equal to P

PRINT specifies the level of printing desired:
0 means error messages only
1 means termination condition messages;
2 means print objective function value after each basis refactorization;
3 means log information at every iteration

PROFIT is part of the problem data structure. It contains the objective function

PTOLER Input for XCHECK: the tolerance for the primal accuracy check

Q The index of the entering variable

R The position of the leaving variable, i.e., the index of the variable leaving the basis is BASIS(R)

REJECT A list of rows that may not be chosen as the pivot row. This is an argument to XDCHR and is part of the PIVOT rejection mechanism for the dual SIMPLEX method

ROWNOS is part of the problem data structure. It contains the row numbers corresponding to the non-zeros in the ACOEFF array

ROWTYP is the array of row types:
+2 means a two-sided constraint;
+1 means less than or equal to;
0 means equation;
-1 means greater than or equal to;
-2 means a free row (functional)

SCODE is a return code for XSTART:
1 means everything OK;
2 means the given basis was singular

START1 is the starting column for the partial pricing procedure

START2 is the last column considered by the partial pricing procedure

STATUS An indicator for each variable
0 means the variable is out at its lower bound;
K means that this is the Kth basic variable;
-1 means that this variable is out at its upper bound;
-2 means that this is a free variable--once in the basis it never leaves;
-3 means that this is an artificial variable--once it leaves the basis it never re-enters
-4 means the variable is locked out of the basis at its lower bound;
-5 means the variable is locked out of the basis at its upper bound
NOTE: Free and artificial variables always have status -2 or -3, respectively; they do not get a positive status when they are in the basis

TERMIN is the termination code for the primal SIMPLEX method:
1 means optimal solution found;
2 means problem is unbounded;
3 means the problem is infeasible;
4 means the presumed optimal solution does not satisfy the accuracy check

THETA is the amount of change in the variable entering the basis, i.e., the value of the winning ratio from the primal ratio test

UCODE is a return code for XUPDAT:
1 means everything is OK;
2 means refactorization suggested because the number of non-zeros has doubled;
3 means the current basis is singular;
4 means storage overflow
Refactorization is imperative if UCODE is 3 or 4.
(The current version of XUPDAT will return UCODE = 1 or stop with an error message)

UPPERB	is part of the problem data structure. It contains the upper bounds on the variables when BNDTYP = 3 or 4
UJ	is used to hold the upper bound on a single variable
UQ	is the upper bound for the entering variable
UZERO	The array containing the values of the dual variables
XBZERO	The array containing the values of the basic variables
YQ	is used to hold the unpacked column that is about to enter the basis
Z	is the value of the objective function
ZNB	is the objective contribution of the non-basic variables

Hierarchical Structure of XMP
(date last modified: August 27, 1980)

Subroutine	Calls
XADDAJ	XDATA1
XADDUB	XDATA3
XBCOMP	XBREDU, XFTRAN, XSTOP
XBREDU	XGETAJ, XGETUB, XSTOP
XBTRAN	LA05B, XSTOP
XCAND	XGETAJ, XDOT
XCHECK	XBREDU, XGETAJ
XCHUZR	XPTIE, XSTOP
XCONSA	NO OTHER ROUTINE
XCONSI	NO OTHER ROUTINE
XDATA1	XSTOP
XDATA2	XSTOP
XDATA3	XSTOP
XDATA4	XSTOP
XDCHQ	XGETAJ, XDOT, XSTOP
XDCHR	XSTOP
XDOT	NO OTHER ROUTINE
XDPH2	XDCHR, XDCHQ (or FDCHQ), XGETAJ, XFTRAN, XUPDAT, XGETUB, XUPDX, XLOG, XFACT, XBCOMP, XBTRAN
XDUAL	XGETAJ, XDOT, XGETUB, XBCOMP, XDPH2, XFACT, XBTRAN, XCHECK, XSTOP
XFACT	XGETAJ, XLA05X, LA05A, XSTOP

XFEAS	XCAND (or FCAND), XDOT, XGETUB, XPIVOT, XBTRAN, XFACT, XBCOMP, XLOG, XSTOP
XFTRAN	LA05B, XSTOP
XGETAJ	XDATA2
XGETUB	XDATA4
XLA05X	XSTOP
XLOG	NO OTHER ROUTINE
XMAPS	XCONSA, XCONSI, XSTOP
XPHAS2	XCAND (or FCAND), XDOT, XGETUB, XPIVOT, XBTRAN, XFACT, XBCOMP, XLOG, XSTOP
XPIVOT	XFTRAN, XCHUZR, XUPDAT, XUPDX
XPRIML	XFEAS, XZCOMP, XBTRAN, XPHAS2, XFACT, XBCOMP, XCHECK, XSTOP
XPRINT	XGETUB
XPTIE	NO OTHER ROUTINE
XSLACK	XADDAJ, XADDUB, XFACT, XBCOMP, XSTOP
XSTART	XGETAJ, XGETUB, XFACT, XBCOMP, XBTRAN, XSTOP
XSTOP	NO OTHER ROUTINE
XUPDAT	LA05C, XSTOP
XUPDX	NO OTHER ROUTINE
XZCOMP	XGETAJ, XGETUB, XSTOP
FCAND	XGETAJ
FDCHQ	XSTOP

Calling Sequences for the XMP Subroutines
(date last modified: June 8, 1981)

```
     CALL XADDAJ (CJ,COLA,COLI,COLLEN,COLMAX, IOERR,J,LENMA,
LENMY,MAPA,MEMORY,N)                                          12 Arguments

     CALL XADDUB (BNDTYP,IOERR,J,LENMA,LENMY,LJ,MAPA,MEMORY,
UJ)                                                           9 Arguments

     CALL XBCOMP (B,BASCB,BNDTYP,BOUND,
     COLA,COLI,COLMAX,
     IOERR,LENMA,LENMI,LENMY,M,MAPA,MAPI,
     S MAXM,MAXN,MEMORY,N,
     STATUS,XBZERO,Z)                                         21 Arguments

     CALL XBREDU (B,BNDTYP,BOUND,
     COLA,COLI,COLMAX,
     IOERR,LENMA,LENMY,M,MAPA,
     MAXM,MAXN,MEMORY,N,STATUS,WORK,
     XBZERO,ZNB)                                              19 Arguments

     CALL XBTRAN (IOERR,LENMI,LENMY,M,MAPI,
     MEMORY,ROW)                                               8 Arguments

     CALL XCAND (BASIS,CAND,CANDA,CANDCJ,CANDI,CANDL,
     COLA,COLI,COLMAX,
     IOERR,LENM,LENMY,
     LOOK,M,MAPA,MAXM,MAXN,MEMORY,N,
     PICK,PHASE,PPRIME,START1,START2,
     STATUS,UZERO)                                            26 Arguments

     CALL XCHECK (B,BASIS,BNDTYP,BOUND,
     COLA,COLI,COLMAX,
     DERROR,DINDEX,DOK,DTOLER,
     PERROR,PINDEX,POK,PTOLER,
     STATUS,UZERO,WORK,XBZERO)                                28 Arguments

     CALL XCHUZR (BASIS,BASLB,BASUB,
     INTYP,IOERR,LQ,M,MAXM,MAXN,N,OUTTYP,
     PIVOT,Q,R,STATUS,THETA,UNBDD,UQ,
     XBZERO,YQ)                                               19 Arguments

     CALL XDCHQ (BETAR,COLA,COLI,COLMAX,
     DFEAS,INTYP,IOERR,LAMBDA,
     LENMA,LENMY,MAPA,MAXM,MAXN,MEMORY,N,
     OUTTYP,PIVOT,Q,STATUS,DUNBDD,
     UZERO)                                                   21 Arguments
```

```
CALL XDCHR (BASLB,BASUB,GR,IOERR,M,MAXM,
NREJ,OUTTYP,R,REJECT,XBZERO)                          11 Arguments

CALL XDOT (COLA,COLI,COLLEN,COLMAX,
MAXM,ROW,ZDOT)                                         7 Arguments

CALL XDPH2 (B,BASCB,BASIS,BASLB,BASUB,
BETAR,BNDTYP,BOUND,
COLA,COLI,COLMAX,
DFEASQ,DTERM,DUNBR,
FACTIT,FACTOR,IOERR,IOLOG,ITER,
LENMA,LENMI,M,MAPA,MAPI,
MAXM,MAXN,MEMORY,N,NTYPE2,PRINT,
STATUS,UZERO,XBZERO,YQ,Z)                             36 Arguments

CALL XDUAL (B,BASCB,BASIS,BASLB,BASUB,
BETAR,BNDTYP,BOUND,
COLA,COLI,COLMAX,
DFEASQ,DTERM,DUNBR,
FACTOR,IOERR,IOLOG,ITER,
LENMA,LENMI,LENMY,
M,MAPA,MAPI,MAXM,MAXN,MEMORY,
N,NTYPE2,PRINT,STATUS,
UZERO,XBZERO,YQ,Z)                                    35 Arguments

CALL XFACT (BASCB,BASIS,BASLB,BASUB,
COLA,COLI,COLMAX,FCODE,
IOERR,LENMA,LENMI,LENMY,M,MAPA,MAPI,MAXM,
MEMORY)                                               17 Arguments

CALL XFEAS (B,BASCB,BASIS,BASLB,BASUB,
BNDTYP,BOUND
CAND,CANDA,CANDCJ,CANDI,CANDL,
COLA,COLI,COLMAX,
FACTIT,FACTOR,FEAS,IOERR,IOLOG,ITER,
LENMA,LENMI,LENMY,LOOK,M,MAPA,MAPI,
MAXM,MAXN,MEMORY,N,NTYPE2,PICK,PRINT,STATUS,
UZERO,XBZERO,YQ,Z)                                    40 Arguments

CALL XFTRAN (COLUMN,IOERR,LENMI,LENMY,M,MAPI,
MAXM,MEMORY)                                           8 Arguments

CALL XGETAJ (CJ,COLA,COLI,COLLEN,COLMAX,
IOERR,J,LENMA,LENMY,MAPA,MEMORY)                      11 Arguments

CALL XGETUB (BNDTYP,IOERR,J,LENMA,LENMY,
LJ,MAPA,MEMORY,UJ)                                     9 Arguments

CALL XLOG (DQ,INTYP,IOLOG,ITER,LEAVE,OUTTYP,
PIVOT,Q,R,THETA,Z)                                    11 Arguments
```

```
CALL XMAPS (BNDTYP,IOERR,LENMA,LENMI,LENMY,
MAPA,MAPI,MAXA,MAXM,MAXN,MEMORY)                    11 Arguments

CALL XPHAS2 (B,BASCB,BASIS,BASLB,BASUB,
BNDTYP,BOUND,
CAND,CANDA,CANDCJ,CANDI,CANDL,
COLA,COLI,COLMAX,
FACTIT,FACTOR,IOERR,IOLOG,ITER,
LENMA,LENMI,LENMY,LOOK,M,MAPA,MAPI,
MAXM,MAXN,MEMORY,N,NTYPE2,PICK,PRINT,STATUS,
TERMIN,UNBDDQ,UZERO,XBZERO,YQ,Z)                    41 Arguments

CALL XPIVOT (BASIS,BASLB,BASUB,
DQ,INTYP,IOERR,LEAVE,
LENMI,LENMY,LQ,M,MAPI,MAXM,MAXN,MEMORY,N,
OUTTYP,PCODE,PIVOT,Q,R,STATUS,THETA,UNBDD,UQ,
XBZERO,YQ,Z)                                        28 Arguments

CALL XPRIML (B,BASCB,BASIS,BASLB,BASUB,
BNDTYP,BOUND,
CAND,CANDA,CANDCJ,CANDI,CANDL,
COLA,COLI,COLMAX,
FACTOR,IOERR,IOLOG,ITER1,ITER2,
LENMA,LENMI,LENMY,LOOK,
M,MAPA,MAPI,MAXM,MAXN,MEMORY,N,NTYPE2,PICK,PRINT,
STATUS,TERMIN,UNBDDQ,
UZERO,XBZERO,YQ,Z)                                  41 Arguments

CALL XPRINT (BASIS,BNDTYP,BOUND,
IOERR,IOLOG,
LENMA,LENMY,M,MAPA,MAXM,MAXN,MEMORY,N,NTYPE2,
STATUS,XBZERO,Z)                                    17 Arguments

CALL XPTIE (CPIVOT,IPIVOT,IWIN)                      3 Arguments

CALL XSLACK (B,BASCB,BASIS,BASLB,BASUB,BLOW,
BNDTYP,BOUND,
COLA,COLI,COLMAX,
IOERR,LENMA,LENMY,LENMY,M,MAPA,MAPI,
MAXM,MAXN,MEMORY,N,
ROWTYP,STATUS,
UZERO,XBZERO,Z)                                     27 Arguments

CALL XSTART (B,BASCB,BASIS,BASLB,BASUB,
BNDTYP,BOUND,
COLA,COLI,COLMAX,IOERR,
LENMA,LENMI,LENMY,M,MAPA,MAPI,MAXM,MAXN,
MEMORY,N,NTYPE2,SCODE,STATUS,
UZERO,XBZERO,Z)                                     27 Arguments
```

```
CALL XUPDAT (IOERR,LENMI,LENMY,M,MAPI,MEMORY,
R,UCODE)                                          8 Arguments

CALL XUPDX (BASIS,BASLB,BASUB,
DQ,INTYP,LQ,
M,MAXM,MAXN,N,OUTTYP,Q,R,
STATUS,THETA,UQ,XBZERO,YQ,Z)                     19 Arguments

CALL XZCOMP (BASCB,BNDTYP,BOUND
COLA,COLI,COLMAX,
IOERR,LENMA,LENMY,M,MAPA,MAXM,MAXN,MEMORY,N,
STATUS,XBZERO,Z)                                 18 Arguments
```

APPENDIX C

UNIVAC RUNSTREAMS

In general, the information contained in this manual pertains to the version of the menu planning model that was placed into operation on the Burroughs B6800 computer at FT Lee, Virginia. Another version of the model, developed at the US Army Concepts Analysis Agency, is in operation on a UNIVAC 1100/82. The two versions are substantially the same, although some differences were incorporated for system compatibility. The following runstreams pertain to the UNIVAC version.

1. DATA HANDLING MODULE

   a. Program file: 75DATAMOD.

   b. Data files:

75RECIPEDAT//AUGUST.	(Unit 10)
75MENUDAT//AUGUST.	(Unit 12)
75MENATTDAT.	(Unit 14)
75TSARECDAT.	(Unit 21)
75WRKRECDAT.	(Unit 22)
75TSANODUPS.	(Unit 23)
75WRKMENDAT.	(Unit 24)

   c. Program collection:

```
@MAP,E ,75ODATAMOD.EXEC
IN 75DATAMOD.EXEC
IN 75DATAMOD.RECIPE
  IN 75DATAMOD.INSREC
  IN 75DATAMOD.LSTREC
  IN 75DATAMOD.LOCREC
  IN 75DATAMOD.DELREC
  IN 75DATAMOD.MODREC
  IN 75DATAMOD.LODREC
    IN 75DATAMOD.INITR
    IN 75DATAMOD.LDTSAR
    IN 75DATAMOD.LDWRKR
IN 75DATAMOD.MENU
  IN 75DATAMOD.INSMEN
  IN 75DATAMOD.LSTMEN
  IN 75DATAMOD.LOCMEN
  IN 75DATAMOD.DELMEN
  IN 75DATAMOD.MODMEN
  IN 75DATAMOD.LODMEN
    IN 75DATAMOD.INITM
    IN 75DATAMOD.LDTSAM
    IN 75DATAMOD.LDWRKM
IN 75DATAMOD.HASHM
```

```
IN 75DATAMOD.HASHR
IN 75DATAMOD.SORTR
IN 75DATAMOD.SORTM
IN 75DATAMOD.MENATT
  IN 75DATAMOD.GETMEN
  IN 75DATAMOD.GETREC
IN 75DATAMOD.PREP
  IN 75DATAMOD.INITMA
IN 75DATAMOD.XREF
  IN 75DATAMOD.LSTXRF
  IN 75DATAMOD.SORTX
IN 75DATAMOD.NMSORT
IN 75DATAMOD.PF
LIB LIB$*FTN10.
LIB LIB$*SORT13.
END
```

d. Program execution:

```
@ASG,A 75RECIPEDAT//AUGUST.
@USE 10,75RECIPEDAT
@ASG,A 75MENUDAT//AUGUST.
@USE 12,75MENUDAT
@ASG,A 75MENATTDAT.
@USE 14,75MENATTDAT.
@ASG,A 75TSARECDAT.
@USE 21,75TSARECDAT
@ASG, A 75WRKRECDAT.
@USE 22,75WRKRECDAT
@ASG,A 75TSANODUPS.
@USE 23,75TSANODUPS.
@ASG,A 75WRKMENDAT.
@USE 24,75WRKMENDAT
@XQT 75DATAMOD.EXEC
```

2. PARAMETERIZATION MODULE

a. Program file: 75PARAMOD.

b. Data files:

75MENATTDAT.	(Unit 14)
75GPDATA.	(Unit 16)
75BOUNDS.	(Unit 17)
75RHS.	(Unit 18)
75XPRIORS.	(Unit 19)

c. Program collection:

```
@MAP,E ,75PARAMOD.EXEC
IN 75PARAMOD.EXEC
  IN 75PARAMOD.MAIDEN
    IN 75PARAMOD.U
    IN 75PARAMOD.MGET
    IN 75PARAMOD.G
    IN 75PARAMOD.R
    IN 75PARAMOD.F
  IN 75PARAMOD.BOUNDS
  IN 75PARAMOD.GOAL
  IN 75PARAMOD.PRIORS
LIB LIB$*FTN10.
END
```

d. Program execution:

```
@ASG,A 75MENATTDAT.
@USE 14,75MENATTDAT
@ASG,A 75GPDATA.
@USE 16,75GPDATA
@ASG,A 75BOUNDS.
@USE 17,75BOUNDS
@ASG,A 75RHS.
@USE 18,75RHS
@ASG,A 75XPRIORS.
@USE 19,75XPRIORS
@XQT 75PARAMOD.EXEC
```

3. SOLUTION MODULE

a. Program file: 75SOLNMOD.

b. Data files:

75RECIPEDAT//AUGUST.	(Unit 10)
75MENUDAT//AUGUST.	(Unit 12)
75MENATTDAT.	(Unit 14)
75GPDATA.	(Unit 16)
75BOUNDS.	(Unit 17)
75RHS.	(Unit 18)
75XPRIORS.	(Unit 19)
Temporary	(Unit 9)
Temporary	(Unit 25)
Temporary	(Unit 26)
Temporary	(Unit 27)
Temporary	(Unit 28)
Temporary	(Unit 30)

c. Program collection:

```
@PREP 75SOLNMOD.
@MAP,E ,75SOLNMOD.XMAIN
IN 75SOLNMOD.XMAIN
IN 75SOLNMOD.POSTPR,.TRMCOD,.GETSOL,.SMENAT
IN 75SOLNMOD.HASHM,.SUMMRY,.LSMEN,.LSGOAL
IN 75SOLNMOD.LSMNRC,.GETMEN,.GETREC,.HASHR
IN 75SOLNMOD.CROSRF,.SORTX,.LSTXRF
IN 75SOLNMOD.LSTHDR,.LSTOBJ
LIB 75SOLNMOD.
LIB LIB$*FTN10.
LIB LIB$*SORT13.
END
```

d. Program execution:

```
@ASG,A 75RECIPEDAT//AUGUST.
@USE 10,75RECIPEDAT
@ASG,A 75MENUDAT//AUGUST.
@USE 12,75MENUDAT
@ASG,A 75MENATTDAT.
@USE 14,75MENATTDAT
@ASG,A 75GPDATA.
@USE 16,75GPDATA
@ASG,A 75BOUNDS.
@USE 17,75BOUNDS
@ASG,A 75RHS.
@USE 18,75RHS
@ASG,A 75XPRIORS.
@USE 19,75XPRIORS
@ASG,T 9.
@ASG,T 25.
@ASG,T 26.
@ASG,T 27.
@ASG,T 28.
@ASG,T 30.
@XQT 75SOLNMOD.XMAIN
```

APPENDIX D

BIBLIOGRAPHY

DEPARTMENT OF THE ARMY

US Army Concepts Analysis Agency (CAA)

Econometric Model for Optimizing Troop Dining Facility Operations (The Army Army Master Menu Study), CAA-SR-82-10, November 1982

MISCELLANEOUS

Knuth, Donald E., The Art of Computer Programing, Vol 3, Addison-Wesley, 1973

Wirth, Niklaus, Algorithms + Data Structures = Programs, Prentiss-Hall, Inc., Englewood Cliffs, NJ, 1976


2-8
DT